ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS.
Universidad Politécnica de Madrid

Proyecto Fin de Carrera

Curso 2013-14

Estudio y desarrollo de un cliente SCEP en una PKI corporativa

Autor:
Rafael Roa Peral

Tutor:
Dr. Carlos Fernández del Val

Fecha de presentación: 27 de Marzo de 2014

A mis padres,
por sus acertados consejos.

Índice general

Índice de figuras

Agradecimientos

El apartado de agradecimientos siempre es uno de los más ingratos de redactar, puesto que siempre se olvida a alguien al que agradecer su contribución (por pequeña que sea) tanto al desarrollo del proyecto como al desarrollo de la carrera.

En primer lugar, he de agradecer a mis padres, Alfonso y Araceli, su paciencia, su cariño y sus consejos, que han sido fundamentales a la hora de seguir mi camino durante mis estudios. Ellos son la razón principal de que haya conseguido el objetivo. También a mis hermanos, Mercedes y Alfonso, que también han supuesto un apoyo importante, aunque no fuera en el ámbito puramente académico del Proyecto.

En segundo lugar, agradezco su contribución a Paula, que me ha ayudado a comprender durante la carrera, que uno solo no siempre consigue los objetivos. Le agradezco estar en los momentos fáciles y en los difíciles. Le debo una parte de mi éxito.

Agradezco al Dr. Carlos Fernández del Val su disponibilidad para dirigirme el proyecto y los consejos y directrices que me ha dado durante su realización.

Agradezco a mis compañeros en la empresa su amabilidad, su disposición a ayudar y que me hayan hecho muy fácil estos meses de trabajo. Gracias a Águeda, Almudena, Jorge, Lorena, Paco, Raquel, Raúl...; y al gerente, Javier, por su disponibilidad y ayuda.

Agradezco a mis amigos en la Facultad: Asus, Bylon, Chema, Java, Kountrix, Jesudas, Nach, Gepeto...; por no ayudarme en el Proyecto y hacerme pasar buenos ratos ratos fuera de él.

Resumen del trabajo

El presente proyecto propone la utilización de un protocolo de obtención y gestión de certificados llamado SCEP (*Simple Certificate Enrollment Protocol*), utilizado inicialmente para el aprovisionamiento automático de certificados en routers y switches de la marca `Cisco Networks`, para su uso en dispositivos personales o corporativos dentro del ámbito laboral.

En la actualidad, se están aplicando nuevas técnicas más eficientes de control del uso de los dispositivos corporativos por parte de los empleados de determinadas empresas. Estas empresas son las encargadas de proporcionar a sus empleados dichos dispositivos, que en muchos casos son un ordenador personal y un teléfono móvil.

En las empresas del sector de las TIC, el uso de esos dispositivos es la principal herramienta de sus empleados, con lo que la seguridad y control son mecanismos fundamentales que deben estar presentes y garantizados. Además, nuevas tendencias como el BYOD (*Bring Your Own Device* o el teletrabajo, permiten a los empleados tanto el uso de los dispositivos corporativos lejos del ámbito laboral (en su domicilio, por ejemplo), como el uso de dispositivos no preparados de antemano por la empresa.

A priori, estas técnicas suponen un avance en comodidad para los empleados; pero puede significar todo lo contrario cuando se habla del mantenimiento de la seguridad y el control de acceso a los recursos de la empresa desde dichos dispositivos (obtención de certificados para acceder a recursos internos, envío de información cifrada entre individuos de la misma empresa, etc).

Es aquí donde cobra sentido la idea de utilizar un protocolo de aprovisionamiento y gestión automatizada de certificados para garantizar la seguridad y control de acceso en la PKI (*Public Key Infrastructure*) corporativa, permitiendo el uso de dispositivos personales, y evitando la acumulación de trámites burocráticos incómodos tanto para el empleado como para la empresa.

Este proyecto plantea la utilización del protocolo SCEP como mecanismo para la realización de la gestión automatizada de certificados ampliando su utilización a dispositivos de escritorio Windows y móviles Android.

Palabras clave: protocolo SCEP, certificado X509, PKI, autenticacion con certificado, obtención automática de certificado.

Abstract

The actual Final Degree Project proposes the usage of a certificate management and supply protocol called SCEP (*Simple Certificate Enrollment Protocol*), on personal or corporate devices, rather than its main use on automatic obtaining of certificates on routers and switches (mainly used by `Cisco Networks`).

Nowadays, new more efficient use-control techniques and methods are being developed to manage and control the use of corporative devices by their owners. Those corporative devices have to be given to employees by their employer company. In most cases, corporative devices are a mobile phone and a notebook.

On ICT companies, the use of corporative devices is the main worktool of their employees. That's why security and use-control are two of the most important facilities to be present and guaranteed. Therefore, new trends as BYOD (*Bring Your Own Device*) or telecommuting, allow to the users, not only to use their own corporative devices outside the physical limits of their companies (i.e: at home), but also to use their own personal devices as non-platformed (by the company) corporative devices.

Theorically, these new techniques bring an important step forward on employees's confort at work, but it can be a difficulty when handling the security or access control to corporate resources from user's devices: from certificate obtention to get access to internal enterprise resources, to sending of encrypted information between employees, etc.

It's here where makes sense the idea of using a protocol that manages the automatic obtention of certificates, to guarantee the security and access control in the corporate PKI, allowing the use of non-corporative devices and avoiding the accumulation of annoying official documents (either for the employer or the employee).

This Final Degree Project proposes the usage of SCEP as automatic certificate-management protocol, extending its usage to Windows desktop devices and Android mobile phones.

Keywords: SCEP protocol, X509 certificate, PKI, certificate-authentication, automatic certificate obtention.

Capítulo 1

Introducción

1.1. Definición del problema

Uno de los aspectos en los que se está incidiendo de manera más notable en el ámbito empresarial en general, y en el de las empresas tecnológicas en particular; es la posibilidad de que los empleados puedan usar sus propios dispositivos tanto dentro como fuera del espacio de trabajo (el primer caso se suele denominar*Bring Your Own Device* ó BYOD por sus siglas en inglés, y el segundo *Telecommuting* ó tele-trabajo) [Bid04].

Esta posibilidad, que repercute favorablemente tanto para la empresa en cuanto a costes (principalmente en la compra de dispositivos para empleados) y para el empleado en cuanto a comodidad (no tiene que portar varios dispositivos dependiendo de si es de uso personal o laboral), conlleva problemas importantes en cuanto a la seguridad de los datos y de las comunicaciones con estos dispositivos [OPR+12].

Actualmente, las empresas se encargan de la compra y distribución de una serie de equipos informáticos y móviles para los empleados que decida. Estos equipos, antes de ser otorgados al usuario final, son plataformados[1]. Por último, son asociados a la identidad del poseedor mediante un contrato de posesión o utilización.

En empresas que hacen uso de certificados digitales para la autenticación de personas y equipos, esto supone un problema adicional, ya que estos certificados pueden caducar o ser revocados antes de la finalización del contrato laboral del poseedor, o antes de la finalización del contrato de asociación poseedor-dispositivo. En ambos casos, se pueden dar varias soluciones: desde la necesidad de llevar el

[1]Dispositivo que dispone del software y personalizaciones preconfiguradas determinadas por las políticas de uso de la empresa (previamente definidas).

dispositivo al encargado dentro de la empresa de la reinstalación de los certificados, a que la empresa tenga definido un sistema de aprovisionamiento de certificados al dispositivo dependiente de las acciones del usuario.

Es en este ámbito en el que se plantea la realización del proyecto: una solución software que permita el aprovisionamiento automático y desatendido de certificados de seguridad, para que los empleados de una determinada empresa puedan utilizar sus dispositivos en cualquier momento y acceder a recursos protegidos de la empresa mediante certificados de seguridad.

Dentro de la situación en la que se desarrolla el proyecto, se parte de la siguiente supuesto: una empresa pública estatal quiere permitir que los empleados puedan consultar su correo electrónico o acceder a las versiones de las máquinas virtuales por Web, utilizando su certificado de autenticación. Para ello, se contrata la realización de dicho proyecto a una empresa tecnológica principalmente dedicada a la ejecución de proyectos informáticos.

El objetivo final pretendido por la empresa cliente, es el desarrollo de una entidad que haga las funciones de Autoridad de Registro (por sus siglas en inglés, RA –*Registration Authority*), que incluya la funcionalidad de un protocolo de gestión automática de certificados; y compatible con la Infraestructura de Clave Pública (por sus siglas en inglés, PKI -*Public Key Infrastructure*) que tienen actualmente en funcionamiento.

Hasta el momento, el proceso que se seguía en dicha empresa cliente para el registro de un dispositivo y su utilización con certificados era el siguiente:

1. El poseedor del dispositivo debía acudir al Operador de Registro (por sus siglas en inglés, RO –*Registry Operator*), el cual recogía la petición y los datos del terminal, y la autorizaba si correspondía.

2. Se enviaba la solicitud al administrador de la Autoridad de Certificación del cliente, que realizaba la emisión de los certificados.

3. El poseedor del dispositivo debía volver al Operador de Registro y solicitar la instalación de los certificados correspondientes en el dispositivo móvil.

Este proceso se repite en el caso en que los certificados sean revocados o caduquen. Este último caso es el que supone el mayor problema, ya que si bien las revocaciones de certificados no ocurren muy a menudo (y suelen ser definitivas, por ejemplo: finalización del contrato de trabajo del empleado), los certificados que se suelen emitir para su uso en algunos dispositivos (por ejemplo, en dispositivos móviles) pueden tener una duración muy corta, con lo que la renovación manual se

convierte en un proceso tedioso y que involucra a más personas aparte del usuario final del dispositivo [OPR+12].

Al inicio del proyecto, la empresa cliente añade un requisito adicional. La empresa cliente ya contaba con una solución software que permitía el aprovisionamiento de certificados en dispositivos de tipo iPAD (ya que el sistema iOS proporciona dicha funcionalidad, aunque no completamente automática), mediante el protocolo de gestión automática de certificados SCEP (por sus siglas en inglés, *Simple Certificate Enrollment Protocol*). Este protocolo ha sido probado con éxito internamente, por lo que se cuenta con un primer protocolo probado en un entorno de *test*[2], y por ello parte con ventaja a la hora de la comparativa con otros protocolos. La empresa cliente quiere una funcionalidad similar a la del protocolo, personalizable y multiplataforma.

1.2. Objetivos del proyecto

Por tanto, dentro de la planificación del proyecto solicitado (desarrollo de la entidad RA con funcionalidad de gestión automática de certificados), se plantea como un primer estudio de viabilidad la ejecución de este Proyecto Fin de Carrera (en adelante PFC), consistente en la decisión de implementación de un protocolo de gestión automática de certificados y el desarrollo de un cliente software en versión escritorio y Android que ejemplifique las bondades del protocolo escogido.

El primer objetivo en la realización del PFC consiste en la presentación de un informe resumen del estado actual de la panorámica de protocolos de gestión automática de certificados, presentando sus características más notables; así como una descripción más detallada del protocolo escogido dentro de ellos.

En segundo lugar, se presenta el diseño a alto nivel de las soluciones software planteadas (versión Android y escritorio), resaltando las decisiones de diseño tomadas, así como las razones que han llevado a ellas. No se presentarán datos o diagramas de implementación a bajo nivel debido a cuestiones de propiedad intelectual y de copyright.

En último lugar, se presentan los resultados obtenidos de ambas versiones desarrolladas, verificando su correcto funcionamiento y destacando aquellas deficiencias o errores (si es que los hubiera), además de los problemas que ha entrañado su desarrollo.

[2]Configuración software y hardware en la que el equipo de pruebas realiza su labor sobre un software en desarrollo.

Capítulo 2

Estado de la cuestión

En primer lugar, y antes de abordar el tema concreto de la gestión automática de certificados en una infraestructura de clave pública (en adelante, PKI), se presenta una panorámica general del ámbito de la seguridad y criptografía de clave pública en una PKI.

2.1. Criptografía simétrica

Los métodos de criptografía simétrica están basados en la utilización de la misma clave para cifrar y descifrar, con lo que emisor y receptor deben conocerla para poder comunicarse de manera segura.

El ejemplo más emblemático en la historia de la criptografía simétrica es el cifrado utilizado por los alemanes en la Segunda Guerra Mundial por medio de su máquina *Enigma*. Esta máquina tenía internamente una serie de conexiones entre los rotores, de forma que cambiaban una letra inicial que se introducía por medio de un teclado por otra establecida por las conexiones de los rotores, de forma que el algoritmo era difícil de averiguar.

El mensaje era cifrado y enviado utilizando una clave diaria (según un diccionario previamente establecido). A la recepción del mensaje, y por medio de la misma máquina en el receptor, era descifrado utilizando la misma clave de cifrado.

El problema que supuso el cifrado que establecía la máquina *Enigma* era el transporte de dichas máquinas (para que tanto emisor como receptor pudieran cifrar y descifrar), y de los libros que establecían las claves del día [Ali09].

Finalmente, por medio de la intervención de criptoanalistas polacos, franceses e ingleses; se pudo averiguar el algoritmo que utilizaba internamente la máquina *Enigma* (mediante el desarme de máquinas interceptadas). Después, haciendo uso de libros de claves diarias capturados durante la guerra, se pudieron descifrar, según cálculos de los servicios de Inteligencia británicos, hasta 84.000 mensajes alemanes por mes desde principios de 1943 hasta el final de la guerra (Mayo de 1945) [Cop].

Los algoritmos de cifrado simétrico más utilizados son los siguientes [EA03, Sta10]:

- DES: Algoritmo de cifrado simétrico por bloques. Actualmente no se usa debido a que es fácilmente rompible por fuerza bruta[1] o por otros tipos de ataque, ya que usa una clave de 56 bits (muy corta).

- 3DES: Aplicación triple del algoritmo DES incrementando la longitud de clave hasta los 112 bits. Usado todavía en algunas transacciones bancarias y en las tarjetas de crédito.

- RC4: Método de cifrado utilizado en el sistema WEP y WPA de cifrado en redes WiFi, o en el protocolo TLS [EA03]. Muy eficiente aunque considerado como *deprecated* (desfasado) por vulnerabilidades encontradas.

- RC5: Método de cifrado por bloques que realiza operaciones de suma modular y XOR en un número de iteraciones. Para determinadas longitudes de claves y parámetros internos, su seguridad ha sido rota por medio de análisis estadístico y matemático [W4].

- AES: Algoritmo definido como estándar por el Gobierno de los Estados Unidos de Norteamérica desarrollado por matemáticos belgas, que realiza cifrado por bloques con varias longitudes de claves. Recientes documentos han revelado que, paradójicamente; los cifrados con AES de 256 y 14 vueltas son más vulnerables que un cifrado AES de 128 bits de clave y 10 vueltas (aunque instituciones como el Departamento de Defensa de los Estados Unidos o *Wikileaks* protegen sus archivos de alto secreto utilizando el primero) [W2, W3].

- IDEA: método de cifrado simétrico por bloques inicialmente utilizado en las primeras versiones de OpenPGP[2]. Poco vulnerable por fuerza bruta (10 $\hat{3}8$ posibilidades de clave), aunque vulnerable a ataques cíclicos (oficialmente roto en 2012).

[1] Ataque que prueba todas las combinaciones de claves hasta encontrar la correcta
[2] Ver PGP

- Blowfish: algoritmo diseñado como reemplazo al antiguo DES y que utiliza cifrado en bloques con vueltas. Considerado débil y reemplazado por AES.

En la actualidad, el problema del cifrado simétrico reside tanto en el análisis del algoritmo (muchos ataques a determinados algoritmos se basan en vulnerabilidades del propio algoritmo de cifrado [EA03]) como en el intercambio de la clave utilizada. Es en este punto donde se incide al querer proteger el intercambio de las claves de cifrado en grupos de comunicación grandes. Se hace necesario algún método de distribución de claves simétricas securizado para evitar ataques, y es ahí donde entra el uso de la criptografía asimétrica.

2.2. Criptografía asimétrica

Como se ha comentado en el punto anterior, uno de los inconvenientes de la criptografía de clave simétrica es la comunicación de la clave compartida de cifrado entre los interlocutores de una comunicación segura, y dificulta en gran manera la distribución de claves a la hora de realizar comunicaciones seguras a través de un canal de comunicación no seguro [AL02].

Por ello, aunque el sistema de criptografía simétrica sea computacionalmente más eficiente (ya que con una clave de menor número de bits se obtiene el mismo nivel de seguridad que con una clave mayor en criptografía asimétrica), esta dificultad de distribución a la hora de realizar comunicaciones par a par seguras entre un número alto de interlocutores hizo que apareciera la criptografía de clave pública.

Durante los años 70 surgieron los primeros métodos de negociación de parámetros para la generación de claves asimétricas. El primer método público de intercambio de claves fue el método Diffie-Hellman, creado por Whitfield Diffie y Martin Hellman [Sta10]. De esta forma, se permitía el intercambio de una clave segura por medio de un canal inseguro sin necesitar un primer secreto compartido.

En la actualidad, se sabe que durante los años 60 y la primera mitad de los años 70, las agencias de seguridad estadounidense (NSA) y británica (MI5) realizaron investigaciones secretas y privadas que llevaron a algoritmos similares al propuesto por Diffie y Hellman, pero éstos fueron los primeros en publicar sus estudios.

El método Diffie-Hellman está basado en la aplicación de una función unidireccional a números escogidos por cada interlocutor con un número primo común. Aplicando exponenciaciones, se conseguía establecer una clave simétrica privada que compartían ambos interlocutores mediante el intercambio en claro por el medio in-

seguro de los números calculados por cada uno (Figura 2.1).

Como complemento a este método de negociación de claves, en el año 1977 se publicó el algoritmo RSA por parte de Rivest, Shamir y Adleman, investigadores del MIT; que constituye el primer algoritmo público de firma y cifrado mediante criptografía de clave pública. Al igual que con el método Diffie-Hellman, un algoritmo similar fue creado unos años antes en el GCHQ británico (Cuartel General de Comunicaciones del Gobierno), dependiente del MI5. Este algoritmo es el más utilizado en la actualidad para cifrar claves simétricas.

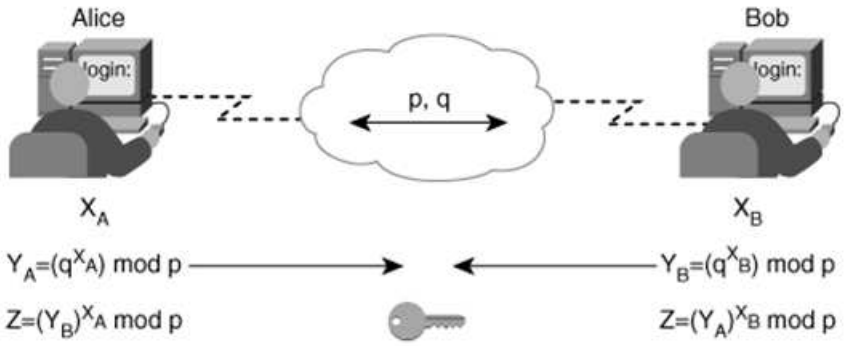

Figura 2.1: Intercambio Diffie-Hellman

Otros algoritmos de cifrado y firma creados posteriormente que se basan en el método de intercambio de claves Diffie-Hellman son:

- El Gamal (1984): algoritmo de firma y encriptado de claves usando como función unidireccional el cálculo de logaritmos discretos. Considerado seguro por no ser posible computacionalmente el cálculo inverso de logaritmos discretos.
- Schnorr (1989): algoritmo de firma que usa como función unidireccional el cálculo de logaritmos discretos. Considerado seguro por no ser posible computacionalmente el cálculo inverso de logaritmos discretos.

Otros algoritmos complementarios al panorama de algoritmos de clave asimétrica o pública son [EA03, Sta10]:

- ECC (*Elliptic Curves Cryptography*) (1985): algoritmo que permite utilizar las estructuras algebraicas de las curvas elípticas como algoritmos de generación de clave pública.

- DSA (*Digital Signature Algorithm*) (1991): algoritmo de firma digital estandarizado por parte del Gobierno de los Estados Unidos de Norteamérica, que incluye generación de claves mediante exponenciación. Existe una variante denominada ECDSA que utiliza criptografía basada en curvas elípticas para la generación de claves.

Los algoritmos asimétricos de generación de claves proporcionan un par de claves asociadas, siendo una pública y otra privada. La aplicación de la función unidireccional permite que se pueda determinar la clave pública mediante la privada asociada, pero no al revés. Esto garantiza que la clave privada no es posible ser generada (el cálculo de la función inversa es computacionalmente imposible en términos de tiempo).

De esta forma se establece una de las claves como privada (y por lo tanto debe permanecer secreta) y una pública, que servirán para realizar cifrado y firma digital. La característica principal es la siguiente: lo que cifra la privada lo descifra la pública, y viceversa.

Las operaciones fundamentales que se realizan utilizando la criptografía asimétrica son [Oso10]:

- Cifra digital: consiste en el encriptado de datos (claves simétricas o estructuras de datos que almacenan claves) con la clave pública del destinatario. Esto garantiza que la información sólo puede ser descifrada con la clave privada del destinatario, y por lo tanto, garantizando la confidencialidad.

- Firma digital: consiste en la obtención de una huella digital o hash de los datos a firmar, y su encriptado con la clave privada del remitente. Con ello se puede comprobar la veracidad del remitente por medio del desencriptado de la firma con la clave pública del remitente (conocido por todo destinatario del mensaje) y comprobando el hash con uno calculado en la recepción. De esta manera, se proporciona integridad (no hay cambios en el mensaje), autenticación (verificación del remitente), y no repudio de origen (el remitente no puede negar su firma).

2.3. Certificados Digitales

Un certificado digital es un documento digital que certifica que la clave pública que contiene pertenece al usuario que identifica.

Concretamente, y según el Instituto Nacional de Estadística: "*Un certificado electrónico es un conjunto de datos que permiten la identificación del titular del certificado, intercambiar información con otras personas y entidades, de manera segura, y firmar electrónicamente los datos que se envían de forma que se pueda comprobar su integridad y procedencia*".

En España, los certificados electrónicos principales reconocidos por entidades públicas son los residentes en el DNI Electrónico (autenticación y cifrado), y los emitidos por la Fábrica Nacional de Moneda y Timbre (FNMT).

El estándar internacional que define un certificado digital reside en la norma UIT-T X.509 [RFC 5280, actualizada por RFC 6818], que establece el formato que debe tener un certificado digital X.509 versión 3 [AL02]. Los datos contenidos en él son los siguientes:

- Versión del estándar X.509 que implementa el certificado.
- Número de serie del certificado.
- Identificador del algoritmo utilizado para el cálculo de la firma.
- Identificación del emisor.
- Fechas de validez
    - Inicio de validez
    - Fin de validez
- Identificación del sujeto propietario del certificado.
- Información de clave pública del sujeto
    - Algoritmo de generación de la clave pública.
    - Clave pública del sujeto.
- Campos opcionales
    - Identificador único de emisor.
    - Identificador único de sujeto.
    - Extensiones
- Firma digital del certificado.

Algunas de las extensiones más usadas o comunes en los certificados digitales son el uso de la clave (establece el propósito de uso de la clave contenida en el certificado: firma, cifra, etc.), uso extendido de clave, o puntos de distribución de Listas de Certificados Revocados (por sus siglas en inglés, CRL -*Certificate Revocation List*).

Esta es la unidad básica de manejo dentro del ámbito de una infraestructura de clave pública, ya que identifica a cada uno de sus miembros así como su asociación a una clave pública.

2.4. Criptografía basada en certificados

La criptografía basada en certificados surge como una particularización de la criptografía basada en identidad [W22] (la cual usa algún dato característico del interlocutor como método de identificación del mismo, como por ejemplo: IP, dirección de email, dirección física, etc.; sobre una clave pública maestra) y apoyada fundamentalmente en la criptografía de clave pública o asimétrica.

En este escenario, cada interlocutor genera su par de claves pública-privada, y dispone de un certificado de una Autoridad de Certificación que usa cifrado basado en identidad. Los interlocutores deben solicitar a la Autoridad de Certificación común que firme sus solicitudes para generar certificados de identidad de cada uno de ellos.

El propósito de los certificados digitales en la criptografía de clave pública depende de la operación que se realiza [Oso10]:

- Cifra: La información a cifrar se encripta con la clave pública residente en el certificado digital del destinatario. Este certificado debe estar en posesión del remitente antes de realizar el cifrado.
- Firma digital: Para comprobar la firma, se ha de utilizar la clave pública del remitente residente en su certificado. El destinatario puede disponer de dicho certificado previamente o recibirlo junto al envío de la información firmada.

De esta forma, cada uno de los interlocutores puede enviar información segura al otro porque posee un certificado firmado por una Autoridad de Certificación común de confianza.

2.5. *Public Key Infrastructure X.509* (PKIX)

2.5.1. Definición

Como complemento a la criptografía asimétrica, y en especial a la basada en certificados, surge la definición de una infraestructura de clave pública o PKI [AL02, Bid04].

Una PKI es un entorno formado por el hardware, software y los métodos y políticas de seguridad necesarios que permiten la ejecución con garantías de operaciones criptográficas de cifrado, firma digital o no repudio de transacciones electrónicas. Es decir, establece la normativa que garantiza la confianza en los elementos que forman la PKI y permiten una comunicación segura y fiable entre ellos.

En particular, se estudiará la infraestructura de clave pública basada en certificados X.509 o PKIX [KTD11]. Esta infraestructura no tiene un estándar propiamente definido, sino que está compuesto por varios estándares del grupo de trabajo PKIX perteneciente al IETF (por sus siglas en inglés, *Internet Engineering Task Force*), como los siguientes:

- Certificate and CRL Profile [RFC 3280]
- Online Certificate Status Protocol [RFC 2560]
- Time Stamp Protocol [RFC 3161]

La definición de la PKI, unido a los algoritmos de clave pública; permiten a los miembros de la misma confiar en que la identidad del interlocutor de la comunicación segura es correcta y verdadera.

La base sobre la que se trabaja en la seguridad en una PKI es la utilización de certificados digitales. Como se ha visto anteriormente, un certificado digital es una asociación entre una clave pública y una identidad, firmada por alguna entidad que garantiza su autenticidad. En una PKI, la entidad encargada de firmar los certificados digitales es una Autoridad de Certificación, en la que todos los dispositivos enrolados en la PKI confían, y por tanto; en todo certificado firmado por ella.

2.5.2. Elementos

Los elementos o entidades principales que conforman una PKI son los siguientes [KTD11, Oso10, dL11, iC11]:

- Autoridad de Certificación: por sus siglas en inglés, *Certification Authority* (en adelante, CA). Como se ha dicho anteriormente, es la encargada de firmar y generar los certificados digitales de los dispositivos que existen y son fiables en una PKI. También se encarga de mantener su propia identidad. Según la firma del certificado de la CA, existen 2 tipos de CA:

  - Si la firma del certificado de CA pertenece a la misma CA (es decir, está firmado con su propia clave privada), se trata de una CA raíz.
  - Si la firma del certificado de CA pertenece a otra CA distinta, se trata de una CA subordinada.

  Otra de sus funciones es el mantenimiento de las listas de certificados revocados dentro del ámbito de la PKI en la que actúa.

- Autoridad de Registro: por sus siglas en inglés, *Registration Authority* (en adelante, RA). Una RA es la entidad encargada de recibir las solicitudes de emisión de certificados para una determinada CA. Debe ser capaz de verificar que todos los datos de la solicitud de emisión del certificado son válidos y pertenecen al agente solicitante. También debe ser capaz de discernir si el agente solicitante puede poseer el certificado digital que está solicitando (dependiendo de las políticas de la PKI)

- Dispositivos cliente: los dispositivos cliente pueden ser clientes hardware o software, que actúan dentro de la PKI en nombre de su poseedor físico. Son los encargados de solicitar la emisión de un certificado a una CA (previamente autorizado por la RA correspondiente) generando una identidad digital del poseedor dentro de la PKI, y de hacer las operaciones que este pretende dentro de una PKI como cifra, autenticación o firma digital de datos.

- Políticas de seguridad y certificación: directivas o normativa a seguir dentro del ámbito que define una PKI: roles y deberes de los diferentes actores dentro de la PKI [OPR+12]. En él se definen puntos principales como por ejemplo:

  - Arquitectura de la PKI
  - Usos de los certificados
  - Nombrado e identificación
  - Generación de claves
  - Procedimientos técnicos o físicos.
  - Otros.

El documento estándar que define los aspectos a tratar por una Declaración de Prácticas y Políticas de Certificación o DPC, es el estándar RFC 3647.

Otros dispositivos o entidades que pertenecen al entorno de una PKI son:

- Autoridad de Validación (*Validation Authority* o VA): encargado de comprobar la validez de certificados digitales.
- Autoridad de Sellado de Tiempo (*Time-Stamp Authority* o TSA): encargado de proporcionar pruebas de tiempo.
- Módulos de Seguridad Hardware (*Hardware Security Module* o HSM): proporciona funcionalidad de generación y almacén de claves criptográficas por medio de un dispositivo hardware.

2.5.3. Funcionalidad

Dentro del ámbito de una PKI determinada se deben ofrecer una serie de funcionalidades básicas como: gestión, comprobación del estado y publicación de certificados digitales.

La gestión de certificados digitales incluye las operaciones a realizar y los trámites especiales que involucran a los certificados dentro de la PKI:

- Renovación de un certificado digital: Solicitud de un nuevo certificado digital con los mismos datos que uno previamente emitido, y que está llegando al final de su vida útil (fecha de caducidad).
- Recertificación de un certificado digital: Solicitud de un nuevo certificado digital con los mismos datos que uno previamente emitido, y que ya ha llegado al final de su vida útil. Este es un caso especial en el que la política de la PKI debe permitir un período de gracia tras la caducidad de un certificado, en el cuál se permite la emisión de un nuevo certificado que amplíe el ya caducado (es decir, sin la generación de nuevas claves por parte del dispositivo o entidad solicitante).
- Revocación de un certificado digital: Solicitud de cese de validez del certificado digital antes de su caducidad. La definición del procedimiento debe venir declarada en la DPC de la PKI. Normalmente se incluye la actuación de una persona física que hace las funciones de Operador de Registro (por sus siglas en inglés: *Registry Operator* o RO), que es el encargado de realizar la solicitud de revocación de un certificado a la CA especificando una razón válida

y comprobable: cese de operación (por ejemplo, finalización de un contrato profesional de un empleado), compromiso de claves (por ejemplo, una clave privada ha sido revelada), etc.

Suele ser el mecanismo más difícil de delimitar mediante un protocolo de gestión de certificados por los problemas de seguridad que entraña con respecto a la clave privada asociada a un certificado [W22].

La comprobación del estado de los certificados se puede realizar por medio de dos métodos en función de las capacidades de la CA (que es la encargada de mantener las listas de revocación de certificados de la PKI):

- Sin conexión: la CA permite la descarga de la lista de revocación de certificados o CRL (*Certificate Revocation List*) por parte del cliente para su consulta posterior sin conexión con la CA.
- Con conexión: la CA proporciona una serie de primitivas de acceso a la comprobación *online* de certificados por parte de la CA, enviando de vuelta al cliente el resultado de la comprobación. Este método es proporcionado por diferentes protocolos, como por ejemplo OCSP (*Online Certificate Status Protocol*).

La publicación de certificados digitales es un servicio básico en una PKI, ya que es la forma que disponen los diferentes clientes de la PKI de acceder a los certificados de sus pares en dicha PKI con los que quieren interactuar de manera segura. El método de publicación de los certificados puede consistir en un simple servicio Web al que se puede acceder libremente, o un directorio activo de certificados a los que los clientes que pertenecen al mismo pueden acceder por medio de sus credenciales.

2.5.4. Protocolos de gestión de certificados

Como se ha visto anteriormente, el apartado de la gestión de certificados es uno de los puntos básicos dentro de los servicios que debe proporcionar una PKI.

Dentro de la panorámica de protocolos que realizan la función de gestión de certificados existe un amplio abanico de alternativas, algunas de las cuales incluyen también funciones de verificación, publicación o comprobación del estado de certificados digitales dentro de la PKI.

Tal y como se analiza a continuación, en todos los protocolos encontrados y expuestos en el estudio, el problema fundamental es el establecimiento de la certificación inicial del sujeto en la PKI. Cada uno de ellos presenta una forma de

autorización inicial por parte de una Agencia de Registro para evitar las certificaciones por parte de entidades ajenas a la PKI, presentando cada una de ellas sus ventajas y desventajas.

Estándar PKCS#10

El estándar PKCS#10 (*Public Key Cryptography Standard 10*) viene definido en los documentos RFC 2986 y 5867, y establece el formato de los mensajes enviados a una CA para solicitar la certificación de una clave pública.

Una solicitud de certificación PKCS#10 incluye la identidad del cliente para su comprobación, así como la clave pública del cliente.

El estándar PKCS#10 no define por sí mismo un protocolo de gestión de certificados, pero es la parte fundamental de los protocolos que realizan gestión de certificados, ya que la solicitud inicial de certificación por parte de un cliente está basada en dicho estándar.

CMP: Certificate Management Protocol

El protocolo estándar CMP establece mecanismos que permiten la creación y gestión de certificados digitales en el entorno de una PKI.

En el momento de la creación del protocolo, los documentos de definición de estándar de PKCS#7 y PKCS#10 eran privados (*RSA Security*), por lo que en la definición del protocolo CMP se incluyeron los mensajes y lógica definida por la definición existente en ese momento de CMP [RFC 2510] y por el protocolo CRMF (*Certificate Request Management Framework*) [RFC 2511], que define un estándar propio de mensaje de solicitud de certificación.

Este protocolo proporciona soporte a los métodos de emisión de certificados por medio de la aceptación de las solicitudes por parte de una RA, así como soporte a pruebas de posesión de claves privadas.

Las desventajas que suponen el uso de este protocolo es la utilización de mensajes no basados en estándares (PKCS#7 y PKCS#10), así como el número de mensajes involucrados en las tareas de gestión de mensajes (hasta 7 tipos de mensajes que implementan las diferentes operaciones del protocolo: solicitud de certificado por parte de una CA, solicitud de certificado por parte de una entidad final, recuperación de clave, etc.), y el uso de un repositorio por parte del servidor en el que almacenar

los certificados y claves (para la implementación de recuperación de credenciales por parte de una entidad final).

Una de las ventajas del protocolo (en sus últimas revisiones) es la posibilidad de solicitar la actualización de clave o una petición de revocación, con lo que se podría considerar un protocolo que tiene más importancia en el lado del servidor que en el del cliente. Otra es la implementación del protocolo por parte de varias soluciones PKI (OpenSSL, Entrust, etc.) y su mejor consideración por parte de entidades como el NIST o el *PKI Forum* [W5].

CMC: Certificate Management over CMS

El estándar CMC viene definido en el documento RFC 2797 (actualizado en su revisión RFC 5272), y especifica un protocolo para la gestión de certificados por medio del estándar CMS, que define un formato de representación de mensajes con firma, cifra o autenticación.

El protocolo definido por CMC se implementa sobre los estándares PKCS#10 y CMS, proporcionando servicios de certificación por medio de pruebas de identidad (*Proofs-of-Identity*) como método de autenticación inicial, y de pruebas de posesión de clave privada (*Proofs-of-Possession*) por parte de una entidad final.

En su última especificación o RFC publicado, el protocolo CMC establece en su parte servidora un servicio de certificación individual. Este consiste en una solicitud de certificación individual de aprobación asíncrona (pudiendo requerir la aprobación manual, o aportación de datos adicionales, según la política de certificación de la PKI), y aunque no se defina explícitamente; servicios de renovación[3] o recertificación[4] de certificados digitales ya emitidos [W5].

Asimismo, proporciona una serie de mensajes de implementación opcional como la petición de revocación de un certificado digital, o la obtención del certificado previamente emitido.

El protocolo proporciona una gestión de claves por parte del cliente, y garantiza una comunicación segura entre la CA y la entidad final, así como entre entidades finales por medio de los certificados obtenidos.

Una de las ventajas de este protocolo es su implementación en la solución de PKI de Microsoft, de forma que el protocolo puede ser fácilmente desplegado en una

[3]Nuevo certificado para el mismo poseedor y clave

[4]Nuevo certificado para el mismo poseedor con nueva clave pública

organización que ya disponga de una PKI basada en Microsoft.

Sin embargo es una solución que está fuertemente ligada a la autorización por parte de una Agencia de Registro en su inscripción inicial de dispositivos en la PKI [W5].

SCEP: Simple Certificate Enrollment Protocol

El protocolo SCEP es un protocolo no estándar definido por la empresa estadounidense Cisco Networks, en principio para los routers y switches de su propia marca; para la gestión automática de los certificados en dichos dispositivos en el entorno de una PKI empresarial [W23].

Su especificación viene definida en un documento *draft* o propuesto como estándar, caducado el 10 de Marzo de 2012 [W25]. Utiliza criptografía basada en los algoritmos RSA.

Proporciona una serie de primitivas de manejo del protocolo en una PKI como la obtención del certificado de la CA, la certificación de un cliente, la renovación de dicho certificado, la renovación previa del certificado de la CA (actuando ante la próxima caducidad), o la consulta del estado del propio certificado del dispositivo.

No proporciona un servicio o método de revocación de certificado dentro de la especificación del protocolo.

Aunque se trata de un protocolo muy específico, con una especificación algo desfasada, se ha ido incluyendo en suites de software PKI posteriores (tanto libres como propietarias: Cisco, Microsoft, SafeLayer, VeriSign, Entrust, OpenSSL, EJBCA, etc.), y su implementación resulta poco costosa debido a que existen varias librerías de libre distribución que lo implementan (SSCEP, EBJCA, jSCEP, etc.)

EST: Enrollment over Secure Transport

Recientemente convertido en estándar (Octubre de 2013) definido en el documento RFC 7030.

Dicho protocolo es similar en funcionamiento y definición a los descritos anteriormente:

- Proporciona métodos de gestión de certificados en una PKI

- Está situado entre la CA y las entidades finales (normalmente junto a una RA)
- Adopta capacidades del protocolo CMP sin usar su sintaxis
- Utiliza como protocolo de transporte HTTP/HTTPS.

Proporciona obtención de certificado de CA por parte de una entidad final, inscripción inicial de una entidad final en la PKI, renovación de certificados o claves y consulta del estado del certificado.

Además proporciona extensiones a los protocolos anteriormente definidos: solicitud de atributos a especificar en la solicitud de certificación y petición de claves generadas en el servidor.

Una ventaja de este protocolo es que no restringe el método de autenticación inicial del cliente en la inscripción inicial, pudiendo aplicar el uso de un secreto compartido, usuario y contraseña, un certificado previamente emitido, etc.

La principal desventaja es que supone un protocolo novedoso, por lo que todavía no está presente en muchas de las soluciones software servidoras de funcionalidad de RA.

Otros protocolos

Existen otros protocolos que implementan la gestión de identidades y claves asociadas al cifrado y firma de documentos, como por ejemplo XKMS (*XML Key Management Specification*) o KMIP (*OASIS Key Management Interoperability Protocol*), más específicos y con una complejidad mayor, con lo que no se han tenido en cuenta en el estudio de la panorámica, excepto únicamente a efectos de conocimiento de su existencia.

2.6. Gestión de certificados en Java

En este apartado se obvian características de la arquitectura de seguridad que utiliza Java, viendo alguna característica destacada de manera superficial, profundizando en el aspecto de manejo de almacenes de certificados y claves.

Desde la versión 5.0 de Java (también llamada *Java 2 Standard Edition*) se incluye el soporte para seguridad y certificados en los paquetes `java.security` (manejo

de estructuras de datos) y `javax.crypto` (proveedor de operaciones e interfaces criptográficas), y no por separado como era hasta ese momento (*Java Cryptography Extension* o JCE, y *Java Cryptography Architecture* o JCA) [W6].

Este soporte proporciona a los desarrolladores y a administradores de seguridad un conjunto de herramientas y APIs que incluyen políticas de seguridad, creación de claves, manejo de claves, verificación de firmas, firma de JAR y otras funcionalidades relacionadas con el manejo y administración de claves.

Un KeyStore es la unidad de almacenamiento de claves y credenciales básica en Java. Se trata de una base de datos que almacena claves y sus certificados de confianza asociados. Las entradas de un KeyStore esán identificadas por un alias propio, y pueden ser de 3 tipos según lo que almacenen:

- Cadena de certificados de confianza
- Certificado de entidad con clave privada asociada
- Secreto compartido.

Un KeyStore puede estar a su vez, protegido por medio de una contraseña maestra que debe conocer el administrador y desarrollador.

Existe una implementación de KeyStore proporcionada por Java, denominada *Java KeyStore* o JKS, para almacenar un conjunto de claves en formato PKCS#8, certificados, y opcionalmente protegido por una contraseña maestra.

Las funcionalidades de seguridad de Java se delegan en proveedores de servicios criptográficos firmados y autorizados por Oracle, siendo los proveedores más importantes: IBMJSSE, SunJSSE, OpenSSL, o BouncyCastle.

2.6.1. Librería *BouncyCastle*

Aunque el API criptográfico de Java es rico y usa el proveedor de servicios criptográficos de Sun, una de las librerías criptográficas más utilizadas en Java es la librería *BouncyCastle*.

BouncyCastle (BC) es un proveedor de seguridad que se puede utilizar en lugar del proveedor de servicios criptográficos predeterminado en Java (dependerá de la versión del *Java Runtime Environment* (o JRE) y proporciona un gran número de algoritmos y mecanismos de seguridad implementando las interfaces definidas por Java [W14].

Las ventajas que supone el uso de esta librería es su licencia de libre distribución, su gratuidad, el gran número de mecanismos de seguridad que implementa, y que se está convirtiendo en una librería que podría ser considerada como estándar de facto por su gran uso en aplicaciones Java y su enorme comunidad on-line.

Recientemente, una versión reducida de los mecanismos implementados en la versión de escritorio de *BouncyCastle* fueron adaptados e incluidos en el sistema operativo Android. Paralelamente surgió una versión adaptada ad-hoc cuyo desarrollo es la adaptación completa de la librería *BouncyCastle* de escritorio a Android. Su nombre es *SpongyCastle*.

2.7. Gestión de certificados en Windows

A continuación se tratan los aspectos más importantes de la gestión de certificados que se realiza en el sistema operativo Windows.

Windows utiliza como almacén de certificados del sistema el registro de Windows. Dentro de las claves \SOFTWARE\Microsoft\SystemCertificates\My\Certificates, existentes en HKEY_LOCAL_MACHINE (los propios de sistema) y en HKEY_LOCAL_USER (los del usuario).

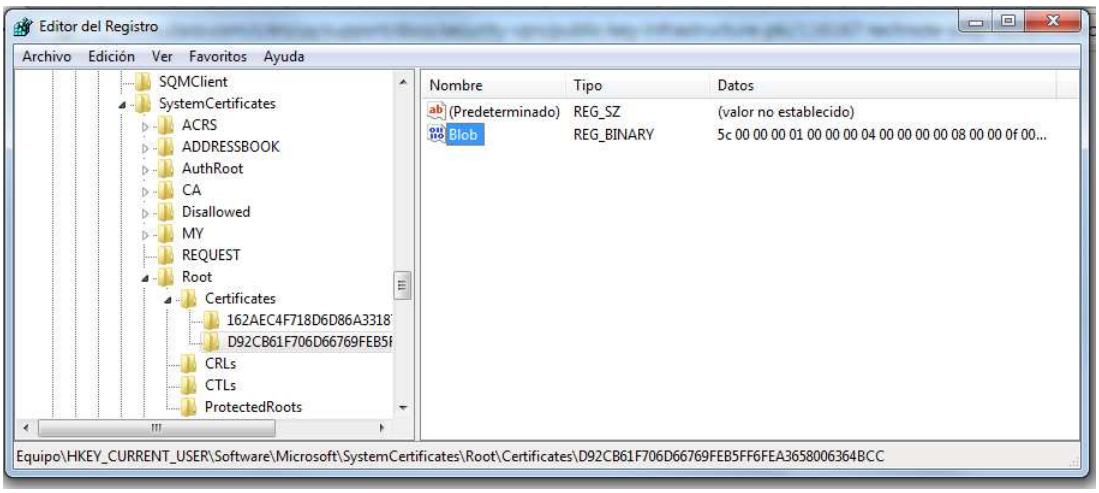

Figura 2.2: Almacén de certificado en Registro Windows

Como se observa en la figura 2.2, se almacena una cadena de bytes que se corresponde al certificado almacenado.

Windows proporciona una serie de herramientas para el control de los almacenes de certificados, por medio de la MMC (*Microsoft Management Console*), que permite ver y gestionar el contenido de los almacenes por medio de interfaz visual.

Asimismo, proporciona una serie de utilidades para el manejo de certificados por medio de comandos, por ejemplo: `certutil`.

2.8. Gestión de certificados en Android

A continuación se tratan los aspectos más importantes de la gestión de certificados que se realiza en el sistema operativo Android.

2.8.1. Android KeyStore

A partir de la versión 4.0 de Android, Android posee un almacén central para gestión de certificados de usuario. Este almacén está controlado por un servicio de SO llamado *keystore_cli* que se encarga de realizar las operaciones de gestión: obtención, consulta y borrado de claves o certificados, y generación de claves simétricas para su uso en aplicaciones (para realizar guardado seguro intradispositivo).

```
root@android:/data/misc/keystore # ls -la
-rw------- keystore keystore       84 2014-03-03 13:12 .masterkey
-rw------- keystore keystore     1428 2014-02-26 15:28 1000_CACERT_BQ_Aquaris_4_5
-rw------- keystore keystore     2020 2014-02-26 15:28 1000_USRCERT_BQ_Aquaris_4_5
-rw------- keystore keystore      804 2014-02-26 15:28 1000_USRPKEY_BQ_Aquaris_4_5
```

Figura 2.3: Vista de directorio - Android KeyStore

Internamente, el almacén guarda su contenido en la ruta `/data/misc/keystore`. Su contenido es parecido al mostrado en la figura 2.3 [W8].

Cada nombre de fichero está compuesto por el UID del usuario que instala el dispositivo (UID 1000 es el usuario `system`), el tipo de entrada del KeyStore (que puede ser CACERT -certificado de CA, USRCERT -certificado de usuario, USRPKEY -clave privada asociada a un USRCERT, y SECRETKEY -secreto compartido de aplicación) [W9].

Se observa que existe una clave maestra (`.masterkey`) del KeyStore. Se encuentra encriptada mediante algoritmo AES de 128 bits con otra clave. La clave de

encriptado se deriva de la clave o método de bloqueo del KeyStore usando el método de derivación de claves PBKDF2 con 8192 iteraciones. El valor *sal* utilizado se genera aleatoriamente y se guarda en el mismo fichero.

Cada fichero que compone las diferentes entradas del KeyStore se guarda con un hash MD5 de la propia entrada, y se encripta con la clave maestra mediante AES de 128 bits [W11].

Estos métodos de seguridad hacen del almacén de Android un sistema seguro para una solución software. Aún disponiendo de acceso *root* al dispositivo, y pudiendo extraer los ficheros correspondientes a las entradas del KeyStore; haría falta adivinar la clave maestra [W8]. Por la definición de la clave maestra anteriormente explicada, se necesitarían las 8192 iteraciones de PKBDF2 para desencriptarlas sobre la clave de desbloqueo del dispositivo; lo cual es demasiado caro en términos de tiempo.

A partir de la versión 4.3 *Jelly Bean*, el sistema operativo Android proporciona la posibilidad de realizar el almacén de claves en un medio hardware. Esta funcionalidad sólo está disponible para unos determinados *System-On-Chip* ó SoC (debido a que disponen de memoria adicional para este propósito) [W9, W10]. Uno de los terminales con esta funcionalidad es el *Google Nexus 4*, que dispone del SoC Qualcom SnapDragon S4 Pro APQ8064.

2.8.2. Métodos de bloqueo en Android

Android establece el uso obligatorio de un método de bloqueo del dispositivo como primera medida de seguridad a la hora de instalar certificados de credenciales de usuario. De esta manera, supone un sistema de seguridad por el que se protege el acceso al almacén central del dispositivo, además de un primer método de acceso físico al dispositivo. [W1]

PIN / Contraseña

El PIN o *Personal Identification Number* es un número de entre 4 y 8 dígitos que debe introducirse en el teléfono cada vez que se quiera desbloquear para su utilización.

El PIN o contraseña que debe establecerse como método de bloqueo debe tener entre 4 y 16 dígitos en el caso del PIN, y entre 4 y 16 caracteres alfanuméricos en el caso de la contraseña. (*Figura 2.4*)

Figura 2.4: PIN/Contraseña en Android

Patrón de desbloqueo

El patrón de desbloqueo consiste en el establecimiento de un camino definido entre puntos de una matriz de dimensión 3x3 (*Figura 2.5*). Existen una serie de recomendaciones relativas a la ejecución del patrón de desbloqueo: no usar patrones que representen figuras relativamente sencillas (como un cuadrado, por ejemplo), no reflejar en la pantalla el trazo del patrón (configurable en Android), o la limpieza de la pantalla para evitar dejar huellas del trazo del patrón.

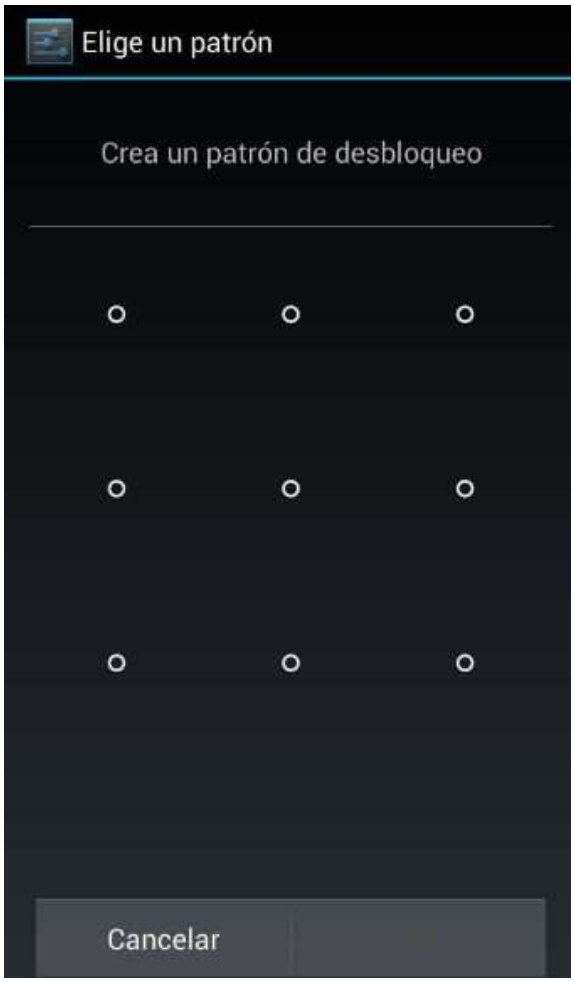

Figura 2.5: Patrón de desbloqueo en Android

Face Unlock

El desbloqueo facial o *Face Unlock* (*Figura 2.6*) es una característica incluída en la versión 4.0 de Android y que supuso una de la características más llamativas de la nueva versión. Aunque en principio suponía una característica de seguridad que acercaba más el control del dispositivo al usuario, ha resultado un agujero de seguridad, puesto que se ha probado que puede ser sobrepasado con una foto de la cara definida como desbloqueo (*bypassing Face Unlock*). De hecho, en el documento de guía de seguridad proporcionado por el CCN (*Centro Criptográfico Nacional*, dependiente del Centro Nacional de Inteligencia) es un método de desbloqueo no recomendado en la seguridad en Android [W1].

Desde la publicación de la versión 4.1 Jelly Bean, dichos problemas de seguridad parecen haber sido solucionados mediante un *Liveness check*[5], aunque sigue siendo fácilmente evitable [W26].

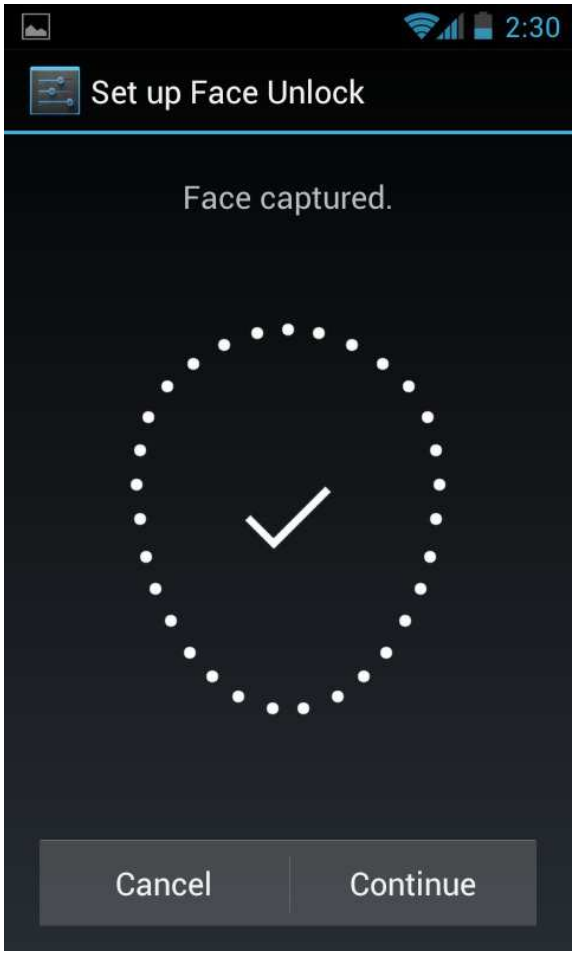

Figura 2.6: *Face Unlock* en Android

2.8.3. Almacén de certificados digitales y credenciales de usuario

Como se ha detallado anteriormente, los almacenes utilizados en las distintas versiones de Android posteriores a la 4.0 utilizan el formato proporcionado la librería de criptografía *BouncyCastle*, que proporciona un API de programación de estándares

[5]Prueba de vida: consiste en detección de parpadeo

criptográficos y de almacenamiento de credenciales. De hecho, las versiones 4.x de Android incluyen dicha librería por defecto dentro del SDK.

Certificados digitales de CA raíz de Android

Desde las versiones iniciales de Android, éste contiene un almacén de certificados digitales de entidades raíz para su utilización, sobre todo, en la autenticación de servidores SSL. Hasta la versión 4.0 de Android, esta lista de entidades no era visible al usuario por medio de la interfaz gráfica del dispositivo, siendo necesario tener permisos de root para acceder al repositorio central y poder añadir o borrar certificados por medio del ADB (por sus siglas en inglés, *Android Debug Bridge*[6]) [W27].

Desde la versión 4.0, esta lista está visible al usuario, pudiendo este inhabilitar los certificados que desee. Un usuario con permisos de *root* y realizando *bypass* sobre el API de gestión del almacén central puede realizar instalaciones de nuevos certificados. También es posible cambiar el repositorio central haciendo uso de la librería *OpenSSL* y *BouncyCastle* creando un repositorio nuevo, e instalándolo en el dispositivo por medio de ADB.

Certificados digitales de usuario

En las versiones de Android 2.x, se disponía de un almacén central de credenciales protegido mediante contraseña, en el que se almacenan contraseñas, certificados digitales, credenciales VPN y credenciales Wi-Fi.

El problema existente es que el API de programación correspondiente a las versiones de Android 2.x-3.x, no proporciona un método de instalación y obtención de certificados digitales, con lo que, para la instalación de certificados digitales era necesaria la inclusión de un servidor Web local en el dispositivo móvil, y realizar la instalación desde el navegador Web de Android, que sí disponía de acceso al almacén de credenciales.

En la versión 4.x de Android sí se ha incluído un API de programación de acceso al almacén de certificados de usuario. Hasta la versión 4.4 (versión actualizada a la redacción del documento), este API sólo proporciona primitivas de instalación y acceso a certificados y claves privadas, no de borrado ni modificación.

[6]Línea de comandos para conectar con un dispositivo Android

Existe un método de borrado de certificados mediante código ejecutado en modo usuario. Este se realiza aprovechando la capacidad *reflection* que proporciona el lenguaje Java (que permite modificar el comportamiento del programa Java en tiempo de ejecución) y la visibilidad del código fuente de Android a través de sus repositorios oficiales [W11].

Mediante este método, es posible copiar el comportamiento de las librerías internas de Java a código usuario y cargarlo en tiempo de ejecución. Esta forma limita el uso de los certificados a la propia aplicación, ya que el acceso a nivel de sistema sólo se puede realizar con permisos de UID `system` o con una aplicación firmada por el fabricante.

Mediante el mismo método, se puede ejecutar el borrado y modificación de los certificados (de igual manera, sólo los relativos a la aplicación).

Si este código es ejecutado en modo *root* o sistema, se puede tener acceso al almacén de certificados completo (con el riesgo de seguridad que ello conlleva).

En cualquiera de las versiones se permite la instalación manual de un certficado desde un KeyStore en formato PKCS#12 (.p12) desde la tarjeta SD del dispositivo. También se permite la instalación de certificados correspondientes a CA usuario por medio de certificados digitales en formato PEM (.cer o .crt).

Capítulo 3

Protocolo SCEP

3.1. Introducción

El protocolo SCEP es un protocolo de gestión automática de certificados digitales que se usa dentro de una PKI, y que está apoyado en los estándares de seguridad PKCS#7 y PKCS#10.

Su especificación se encuentra definida en un *draft* de la IETF (*draft-nourse-scep-23*), originalmente desarrollada por la empresa estadounidense de telecomunicaciones Cisco, y que actualmente se encuentra caducado desde fecha de Marzo de 2012 [W25].

A pesar de no ser un estándar y estar caducado desde hace casi 2 años, se sigue desarrollando sobre él, y empresas importantes del sector de la seguridad en TI siguen implementando el cliente y servidor del protocolo en nuevas versiones de sus plataformas (SafeLayer, por ejemplo).

Se distinguen 2 entidades fundamentales en el protocolo:

- Cliente (*Requester*): por ejemplo, un cliente IPSEC o un servidor gestor de certificados de máquinas asociadas.
- Servidor (*Server*): normalmente irá junto a una RA, y puede ir junto a una CA o por separado.

El protocolo proporciona una serie de operaciones que permiten la gestión de certificados de manera automática entre un cliente y un servidor SCEP (que normalmente irá asociado con una RA).

Las características principales del protocolo son [W23]:

- Modelo petición-respuesta basado en HTTP
- Soporta solamente criptografía basada en RSA
- PKCS#10 como formato de solicitud de certificación
- PKCS#7 como formato de mensaje encriptado/cifrado.
- Solicitud asíncrona de certificados.
- Soporta la consulta limitada de listas de revocación (CRL)
    - No soporta la consulta online del estado de certificados (OCSP).
- NO soporta la solicitud de revocación
- Requiere, como método de seguridad y autenticación adicional; el uso de un desafío secreto compartido (*challenge password*) en las solicitudes de certificación.

El flujo de trabajo normal entre cliente y servidor es el siguiente:

1. El cliente SCEP obtiene una copia del certificado de la CA, y lo valida mediante algún método de comprobación (hash preprovisionado, por ejemplo).
2. El cliente genera una solicitud de certificación y la envía de manera segura al servidor SCEP.
3. El servidor SCEP recibe la solicitud, y una vez comprobada, la envía a la CA para la emisión del certificado correspondiente.
4. El cliente SCEP debe esperar a que el servidor SCEP le responda, preguntándole acerca del estado de la solicitud periódicamente.
5. El servidor SCEP responde con el certificado emitido por la CA y lo reenvía al cliente SCEP solicitante.
6. El cliente SCEP remite una nueva solicitud de renovación cuando su certificado actual esté próximo a la caducidad.
7. El cliente SCEP pregunta al servidor SCEP acerca del estado de su certificado actual mediante la petición de CRL o acceso a los CDP's que mantenga su CA asociada.

Este flujo de trabajo puede sufrir modificaciones debido a que se puede incluir conexión mediante HTTPS, las solicitudes pueden ser rechazadas, o el cliente puede sufrir cambios en las estructuras que mantiene (de forma que tiene que restaurarse a un estado estable, bien mediante la desinstalación de su información y la solicitud de nuevos certificados; o mediante la solicitud de los certificados mediante la petición al servidor SCEP).

3.2. Formato de mensaje SCEP

3.2.1. Formato general: *pkiMessage*

El tipo de mensaje que se intercambia el cliente y servidor SCEP es un PKCS#7 firmado con la clave privada del emisor y que tiene la siguiente estructura (sobre los campos de un PKCS#7 estándar) [W23]:

- Version = 1
- ContentType = pkcs-7 1 [Data]
- SignedContent (si está presente)
  - pkcsPKIEnvelope (PKCS#7 firmado)
- SignerInfo: conjunto de authenticatedAttributes (atributos de información de la transacción)
  - transactionID
  - messageType
  - senderNonce
  - pkiStatus (si es respuesta)
  - recipientNonce (si es respuesta)

3.2.2. Formato de petición SCEP

El protocolo SCEP utiliza mensajes de tipo HTTP GET para solicitar información o realizar peticiones a la CA. El formato de dichos mensajes HTTP es el siguiente:

```
GET CGI_PATH CGI_PROG ?operation= OPERATION &message= MESSAGE
```

Donde los campos en rojo significan:

- CGI_PATH: define la URI de invocación del programa CGI que realiza un *parse* de la solicitud.
- CGI_PROG: ruta del programa que la CA utiliza para administrar las peticiones y transacciones SCEP. Este campo puede ser ignorado y utilizar sólo CGI_PATH (por ejemplo la implementación de SCEP de Microsoft, llamada *Network Device Enrollment Service* –NDES).
- OPERATION: nombre del tipo de mensaje que se envía, y depende de la fase o estado de la transacción SCEP que se esté realizando.
- MESSAGE: mensaje que se envia. Normalmente consiste en un mensaje PKCS#7 codificado en base 64, aunque puede ser un mensaje ASCII sin codificar.

El protocolo admite también el envío de mensajes de tipo HTTP POST, si el servidor SCEP lo permite. En este caso, el campo MESSAGE irá incrustado en la petición HTTP y codificado en binario.

3.2.3. Formato de respuesta SCEP

Las respuestas del protocolo SCEP son devueltas como contenido HTTP estándar, con un valor de atributo *Content-Type* correspondiente al tipo de respuesta y directamente relacionado con la petición original a la que responde. El contenido se devuelve en binario, y no en Base 64 como en la petición que origina la respuesta.

Posibles valores de *Content-Type*:

- *Application/x-pki-message*: respuesta a las peticiones PKCSReq, GetCertInitial, GetCert o GetCRL.
- *Application/x-x509-ca-cert*: respuesta a la petición GetCACert.
- *Application/x-x509-ca-ra-cert*: respuesta a la petición GetCACert cuando hay una RA intermedia.
- Application/x509-next-ca-cert: respuesta a la petición GetNextCACert.

3.3. Entidades SCEP

3.3.1. Cliente

El cliente SCEP se identifica con el dispositivo que realiza las peticiones SCEP contra el servidor SCEP [W23].

Antes de realizar cualquier transacción sobre una PKI, el dispositivo cliente debe poseer un par de claves pública y privada asociadas (en el *draft* actual -23, sólo está permitido el algoritmo RSA), que usará para firmar los mensajes SCEP y asegurar la comunicación con la CA.

El cliente debe tener configurado además previamente el nombre de dominio o IP de la entidad servidora de SCEP, así como el path CGI de ejecución de SCEP en el servidor. Si la CA requiere de autenticación en el cliente, deberá tener también aquella información necesaria para su efecto.

3.3.2. Servidor

El servidor SCEP se debe ver desde dos perspectivas: CA SCEP y RA SCEP, dependiendo de dónde se sitúe el servidor SCEP: junto a la CA o junto a la RA [W23].

CA SCEP

Una CA SCEP es una entidad que firma certificados de cliente. El nombre de la CA firmante debe aparecer en el campo *issuer* (emisor).

Antes de que pueda ocurrir cualquier operación de PKI, la CA SCEP obtiene un certificado de CA que cumple el estándar de *RFC 5280*. Este certificado debe ser emitido por una CA de mayor nivel.

El certificado de servidor CA SCEP debe ser obtenido al margen del cliente SCEP. El hash del certificado CA debe ser usado para autenticar un certificado CA distribuido mediante la respuesta de *GetCACert*. El hash es creado calculando un SHA-1, SHA-256, SHA-512 o MD5 sobre el certificado CA SCEP.

La CA debe incluir también en cada certificado que emita o respuesta cRL, una extensión *cRLDistributionPoint*; en cuyo caso debería estar online en todo momento.

Si un cliente ha recibido ya un certificado identificado por su par de claves, el servidor debe retornar ese mismo certificado ante una nueva petición. Está prohibido que el cliente pueda publicar cualquiera de los campos de la petición de certificación, excepto la clave pública, que será la misma que en el certificado emitido.

Si un cliente no recibe respuesta a una petición pendiente por un tiempo determinado, puede resincronizarse mediante la reemisión de la petición original con el asunto original, nombre e ID de transacción. La CA debe retornar el estado de la transacción original, incluyendo el certificado si este ha sido otorgado. La CA no debe crear una nueva transacción a no ser que el certificado original haya sido revocado o una transacción llegue más allá de la mitad del período de validez del certificado.

RA SCEP

Una RA SCEP es un servidor SCEP que realiza comprobaciones de validación y autorización del cliente SCEP pero remite las peticiones de certificación a la CA. El nombre de la RA no aparece en el campo *issuer* de los certificados resultantes.

La RA debe retornar el certificado de RA junto con el certificado de CA en la respuesta *GetCACert*. La existencia de un certificado RA en esta respuesta le indica al cliente la existencia de una RA. Para realizar una comunicación segura con una RA, el cliente especificará la RA como el destinatario de los siguientes menajes SCEP.

Para poder servir las peticiones de certificados, la RA debe pasar las peticiones al servidor CA para su firma. La RA debe usar SCEP para comunicarse con la CA, en cuyo caso, actúa como cliente en la comunicación con el servidor SCEP CA; y como servidor en la comunicación con el cliente SCEP.

La RA debe responder a las peticiones de certificado por parte del cliente con *PENDING* mientras esté en comunicación con la CA (por ejemplo, si la CA debe autorizar manualmente una petición de certificación , y retorna *PENDING* a la RA, ésta debe responder lo mismo al cliente peticionario mientras siga esperando a la CA).

La comunicación entre la RA y la CA puede realizarse sobre otros protocolos como *Certificate Management over CMS* [*RFC 5272*].

3.4. Operaciones principales

El protocolo SCEP proporciona una serie de operaciones principales que deben ser soportadas por el servidor y cliente SCEP

3.4.1. Autenticación de CA

El protocolo SCEP usa el certificado de la CA (o el de la RA en su defecto) para asegurar el intercambio de mensajes para la solicitud inicial de inscripción del cliente.

Por ello, el cliente SCEP necesita como primer paso del protocolo la obtención de una copia del certificado de la CA. La operación del protocolo asociada se denomina **GetCACert**. Su mensaje asociado es el siguiente:

```
GET/POST CGI_SCEP ?operation= GetCACert &message= MESSAGE
```

En este caso, el mensaje es opcional, aunque se deja a la implementación realizada por el servidor SCEP. Si es necesario, el campo message debe contener el identificador de la CA de la cual se quiere obtener el certificado (en caso de existir una jerarquía de CA).

La respuesta a este mensaje es un PKCS#7 de tipo Degenerate certificates-only, que contiene la cadena de certificados que contiene los certificados de CAs y RAs asociadas.

3.4.2. Inscripción de cliente

El cliente SCEP envía la petición HTTP, cuyo formato se asemeja al siguiente:

```
GET/POST CGI_SCEP ?operation= PKIOperation &message= MESSAGE
```

En este caso, el campo *message* es obligatorio, y debe contener el PKCS#7 codificado en base 64 que se envía al servidor SCEP, que contiene el PKCS#10.

El esquema de generación del PKCS#7 de envoltura de petición de certificación es el siguiente (Figura 3.1):

1. Se crea el PKCS#10 con la clave pública del cliente y el *challenge password* obtenido.
2. Se crea un PKCS#7 que contiene el PKCS#10 encriptado con la clave pública de la CA/RA.
3. Este último PKCS#7 se incluye dentro de otro PKCS#7 firmado. El certificado utilizado para la firma puede ser (dependiendo de la política de la CA):
   - Certificado ya instalado en el dispositivo (de otra CA, de otro emisor autorizado o uno emitido por la CA objetivo previamente.)
   - Certificado auto-firmado.
4. Se codifica en base 64 convirtiendo los caracteres no representables en una URL, y se introduce la lista de caracteres en el campo *message* de la petición.

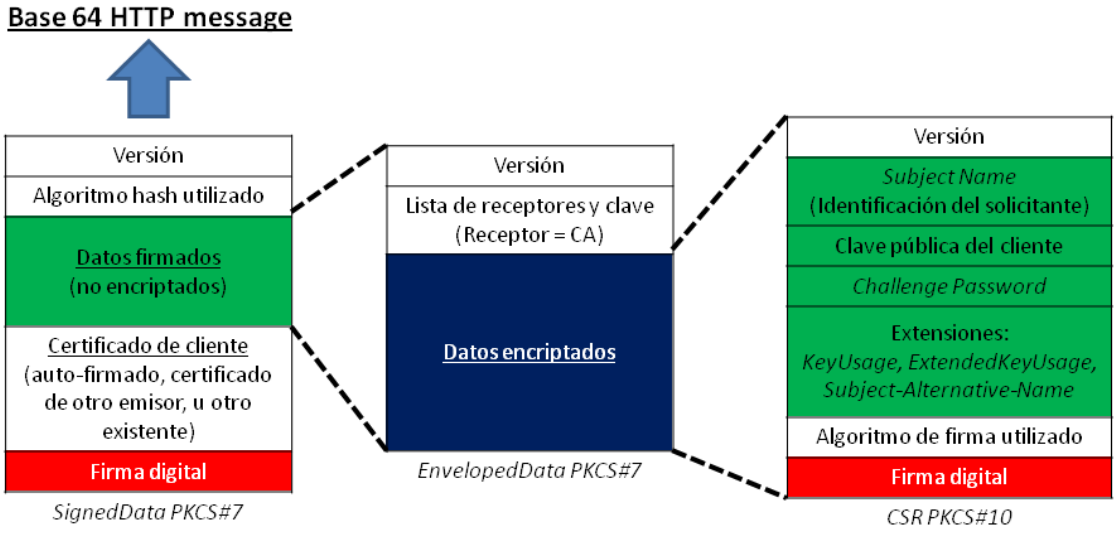

Figura 3.1: Esquema de petición de certificación

La respuesta del servidor SCEP puede ser:

- Rechazada (*Reject*): no incluye ningún certificado, e incluye la razón que esgrime la CA para rechazar la solicitud de certificación (tamaño de clave no válido, *challenge password* no válido, petición no valida, etc.)
- Pendiente (*Pending*): El administrador de la CA debe aprobar o rechazar la solicitud de manera manual, y en el momento no lo ha hecho todavía.

- Aprobada (*Success*): La petición está bien formada y la CA ha aceptado la solicitud. Se devuelve un PKCS#7 firmado por la CA que contiene un PKCS#7 encriptado con la clave pública del cliente solicitante, y que a su vez contiene un tipo especial de PKCS#7 denominado *Degenerate Certificates-only* PKCS#7, que contiene una lista de certificados o CRLs (Figura 3.2).

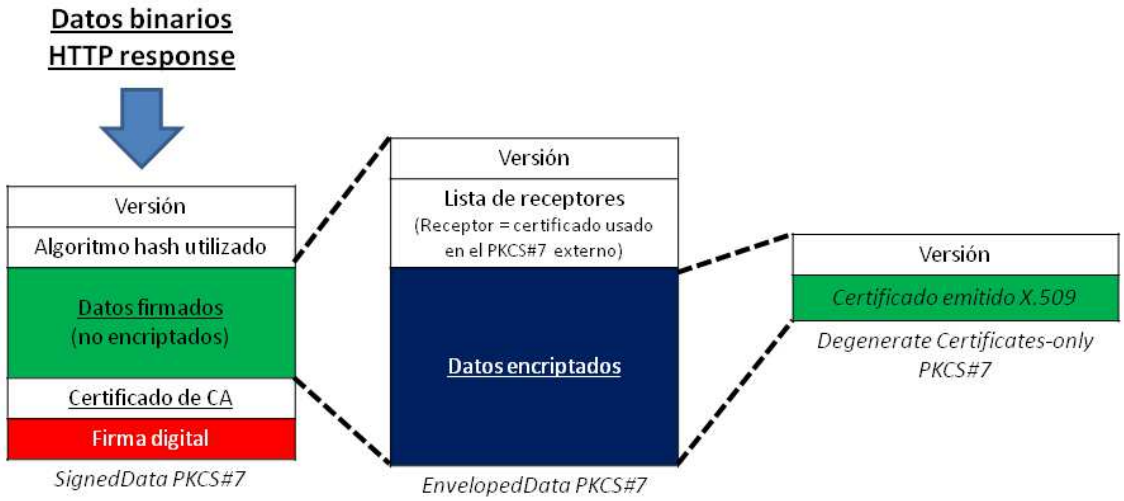

Figura 3.2: Esquema de recepción de petición aceptada

3.4.3. Reinscripción de cliente

La reinscripción de un cliente puede darse en dos circunstancias:

- Renovación de certificado de cliente: también denominada *Renewal*. Se debe realizar unos días previos a la caducidad del certificado de cliente. La CA realizará una renovación previa del certificado del cliente, y el cliente debe ser el encargado de mantener ambos certificados y de realizar el cambio del instalado cuando el previo caduque.

  Los campos *issuer* y *subject* deben coincidir (Mensaje *IssuerSubjectAndName*).

- Renovación de certificado de CA: también denominada *Rollover*. Se debe realizar unos días previos a la caducidad del certificado de CA. La CA realiza una renovación previa de su certificado y de los que ha emitido, de forma que el cliente debe ser el encargado de solicitar ambos certificados renovados, y de cambiar los instalados por los renovados cuando el certificado de la CA actual caduque.

Ambos métodos de reinscripción de cliente deben estar soportados por el servidor SCEP y la CA, ya que si no; toda renovación debe realizarse como si fueran solicitudes nuevas (con la consiguiente desinstalación de los correspondientes certificados en el dispositivo cliente).

3.5. Operaciones secundarias

3.5.1. Obtención de capacidades de la CA

El protocolo SCEP usa un tipo de mensaje especial (**GetCACaps**) para preguntar al servidor SCEP por las capacidades de la CA. Su mensaje asociado es el siguiente:

```
GET/POST CGI_SCEP ?operation= GetCACaps &message= MESSAGE
```

En este caso, el campo *message* es opcional. Su significado depende de la implementación del servidor SCEP (pudiendo significar el ID del dispositivo, o el de la CA de la cual se quiere obtener las capacidades).

Mediante este tipo de mensaje, el cliente SCEP sabe qué operaciones principales o secundarias no puede realizar por no estar soportadas por la CA y el servidor SCEP, o qué algoritmos de seguridad debe usar para estar en concordancia con la CA.

El mensaje de vuelta certifica si la CA es capaz de:

- Admitir pre-renovación de certificado de la CA (*Rollover*)
- Admitir mensajes HTTP POST
- Admitir pre-renovación de certificado de cliente (*Renewal*)
- Admitir algoritmo de hash SHA-512
- Admitir algoritmo de hash SHA-256
- Admitir algoritmo de hash SHA-1
- Admitir encriptado Triple-DES.

Por tanto, el cliente debe soportar todas las capacidades de la CA y adaptarse a ellas. Si no admitiera ningún algoritmo de hash SHA, debe usar MD5.

3.5.2. Obtención de CRLs

El protocolo SCEP usa un tipo de mensaje especial (**GetCRL**) para que un dispositivo cliente pueda preguntar a la CA por el estado de su certificado. Este mensaje puede no estar soportado por el servidor SCEP, en cuyo caso, debería procederse según 2 métodos:

1. Acceder a los CDPs mantenidos por la CA.
2. Acceder al OCSP mantenido por la CA.

La especificación del protocolo recomienda que no se soporte este método por la misma razón que no recomienda el acceso por OCSP, ya que convierte a la CA en un servicio de alta disponibilidad.

3.6. Microsoft NDES

La implementación del servidor SCEP realizada por Microsoft se denomina NDES (*Network Device Enrollment Service*) [W24]. Aunque la implementación de Microsoft en Windows Server 2008 sigue la definición del *draft* (versión número 18) del protocolo [W24], tiene alguna serie de particularidades:

- Se ejecuta realizando funcionalidad de RA de Microsoft, y en su forma de instalación más sencilla, se instala junto a la CA de Microsoft.
- Posee un panel de administración (*mscep_admin*) que permite al Administrador de SCEP proporcionar al cliente SCEP el hash del certificado de la CA en formato MD5, así como el *challenge password* a proporcionar al administrador del dispositivo a inscribir en la PKI [W24].
- No posee algunas de las características del protocolo SCEP, aunque muchas de ellas han ido añadiéndose por medio de la publicación de *hotfix*[1] adicionales:
  - No dispone del mensaje **GetCACaps**. (Solucionado en hotfix KB2483564)
  - No realiza la renovación automática por no disponer del tipo de mensaje denominado *IssuerSubjectAndName* (Solucionado en hotfix 959193).

[1] Parche que soluciona una deficiencia en un programa

- Microsoft no documenta acerca de la gestión del *CA RollOver*, con lo que se supone que no implementa dicho tratamiento. De hecho, la solución documentada por Microsoft ante una renovación o revocación de certificados de CA es la reinstalación de dichos certificados en el dispositivo, así como la reinscripción del mismo en la PKI.

En su funcionamiento normal, la lógica de funcionamiento y que se debe seguir para inscribir un dispositivo en la PKI por medio del protocolo, se detalla a continuación (Figura 3.3):

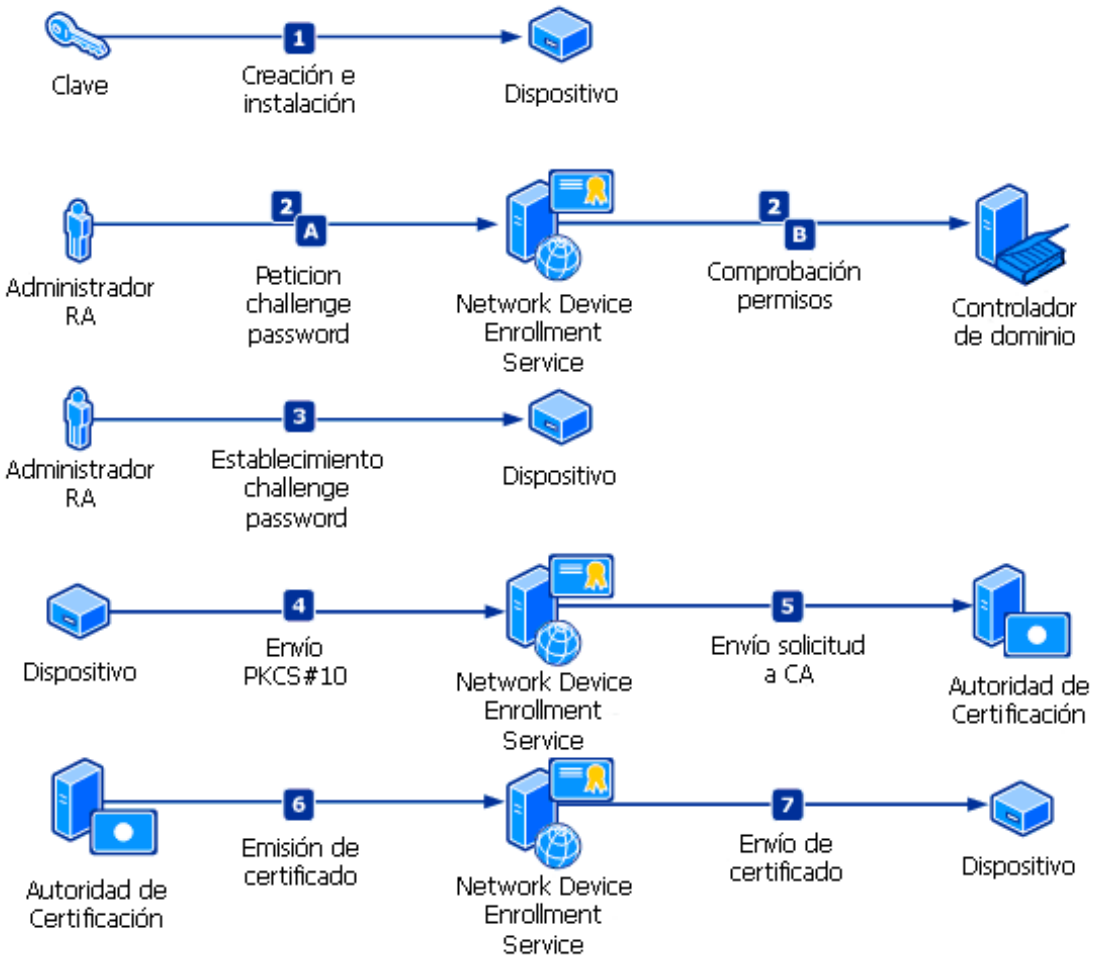

Figura 3.3: Funcionamiento del protocolo NDES

Paso 1. Introducir o proporcionar un par de claves pública-privada al dispositivo (el

método de generación y obtención de dicho par de claves queda fuera del protocolo).

Paso 2. El Administrador del servidor SCEP, por medio de sus credenciales de dominio; pide una OTP (*One-Time Password*) que se usará como *challengePassword*. (Usando la aplicación MSCEP_ADMIN de Windows - Figura 3.4)

Paso 3. El Administrador introduce dicho challengePassword en el dispositivo para poder realizar la inscripción.

Paso 4. Se arranca el cliente SCEP residente en el dispositivo, el cual, utilizando los datos introducidos anteriormente, realiza la petición de certificación y la envía al servidor SCEP.

Paso 5. El Servidor SCEP realiza la comprobación de los datos y envía la petición de certificación a la RA.

Paso 6. La CA emite el certificado hacia el servidor SCEP.

Paso 7. El servidor SCEP envía el certificado al dispositivo solicitante para su uso.

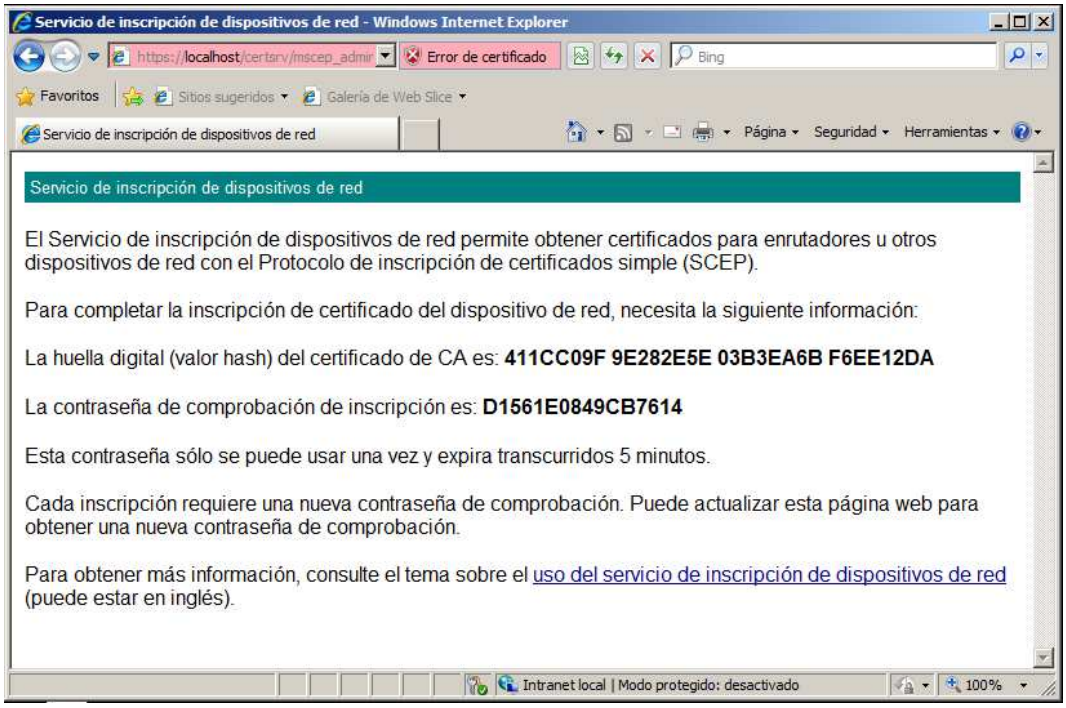

Figura 3.4: Interfaz Microsoft SCEP Administration - *mscep_admin*

El rol de servicio NDES se ejecuta en conjunto con el servidor Web de Microsoft (*Internet Information Service* o IIS), para ofrecer a través de un directorio virtual, el path CGI HTTP de conexión de los clientes SCEP de los distintos dispositivos confiables de la PKI; y en conjunto al Servicio de Directorio Activo (*Active Directory Certificate Service* o AD:CS).

3.7. Cliente SCEP: BlackBerry

Los dispositivos con sistema operativo BB 10, disponen de la funcionalidad del protocolo SCEP a través de la aplicación BlackBerry Device Service.

El sistema funciona de la siguiente manera [W20]:

1. Se ha de instalar la aplicación BlackBerry Device Service.
2. Hay que crear un usuario en el sistema BlackBerry Device Management y asociarlo a un dispositivo BB.
3. Se configura en la aplicación BlackBerry Device Service (Figura 3.5) el servicio de cliente SCEP.
4. En el dispositivo BB asociado a la cuenta de usuario, se registra dicho usuario en el servidor (Figura 3.6).
5. Se ejecuta el cliente SCEP del smartphone.

El smartphone con BlackBerry 10 realiza una solicitud CSR PKCS#10 con la identidad del usuario y la envía al servicio BlackBerry Device Service. Éste comprueba la información proporcionada por el dispositivo con el usuario previamente registrado. Si es correcta, reenvia el CSR al dispositivo con la información necesaria del servidor SCEP. Por último, el smartphone recoge dicha información e interactúa directamente con la CA.

Como se puede observar, la configuración de SCEP no reside propiamente en el dispositivo, sino en el servicio de BlackBerry Device Service. Esto puede significar una ventaja en cuanto al control de dispositivos y seguridad, y desventaja en cuanto a que se incluyen unos primeros pasos de autenticación no necesarios en el protocolo SCEP.

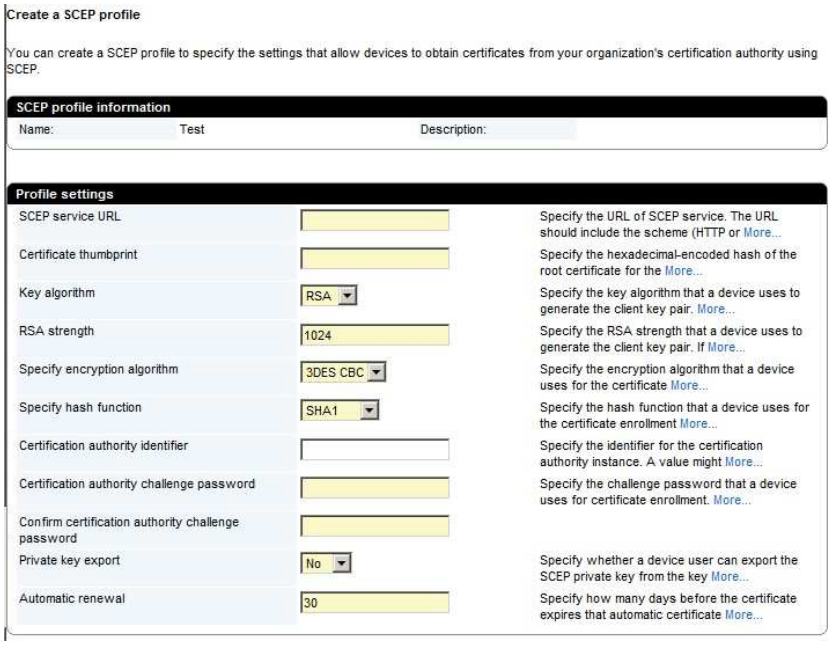

Figura 3.5: BlackBerry Device Service - SCEP

Figura 3.6: BlackBerry Device Service - Login en dispositivo

3.8. Cliente SCEP: iOS

Los dispositivos con sistema operativo iOS como iPod, iPad o iPhone a partir de la versión iOS 5, disponen de la funcionalidad del protocolo SCEP a través de la aplicación iPhone Configuration Utility [W17, W18, W19].

El comportamiento del cliente SCEP en iOS es análogo al especificado en BlackBerry, actuando la aplicación iPhone Configuration Utility como iniciador de la transacción de *enrollment* del dispositivo iOS con el servidor SCEP:

1. Requiere de una primera inscripción en la aplicación iPhone Configuration Utility del dispositivo a enrolar en la PKI mediante SCEP (Figura 3.7)

2. Se configura la conexión con el servidor SCEP por medio de dicha aplicación (Figura 3.8)

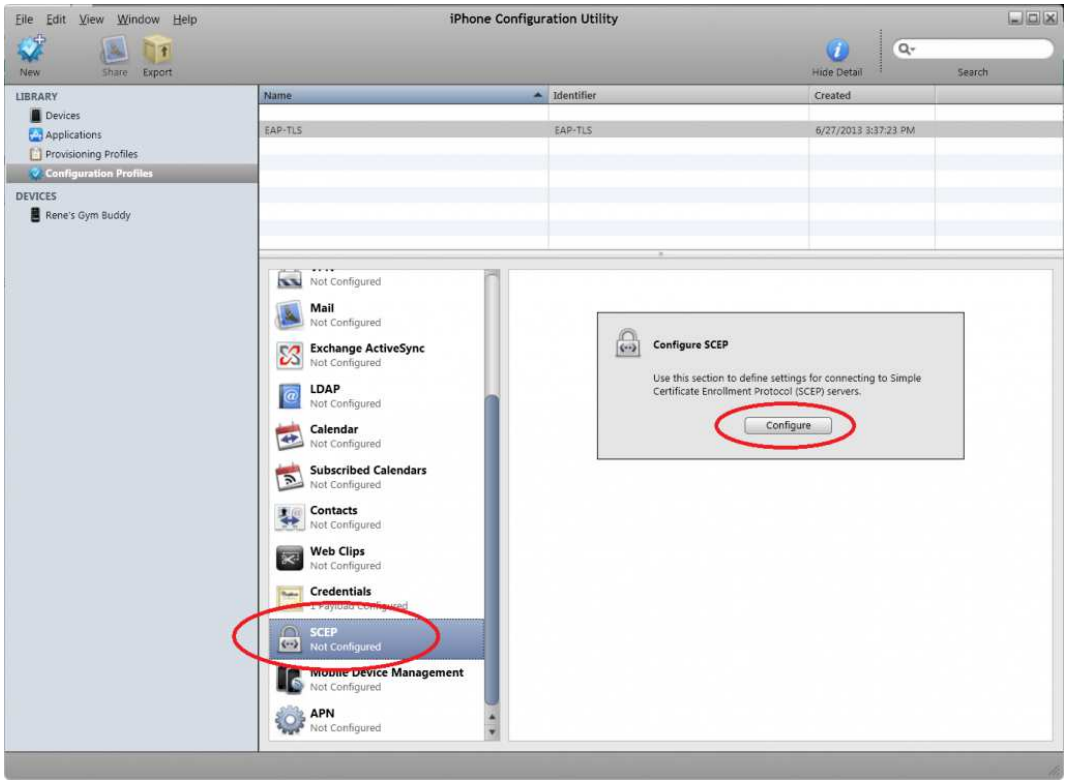

Figura 3.7: iPhone Configuration Utility

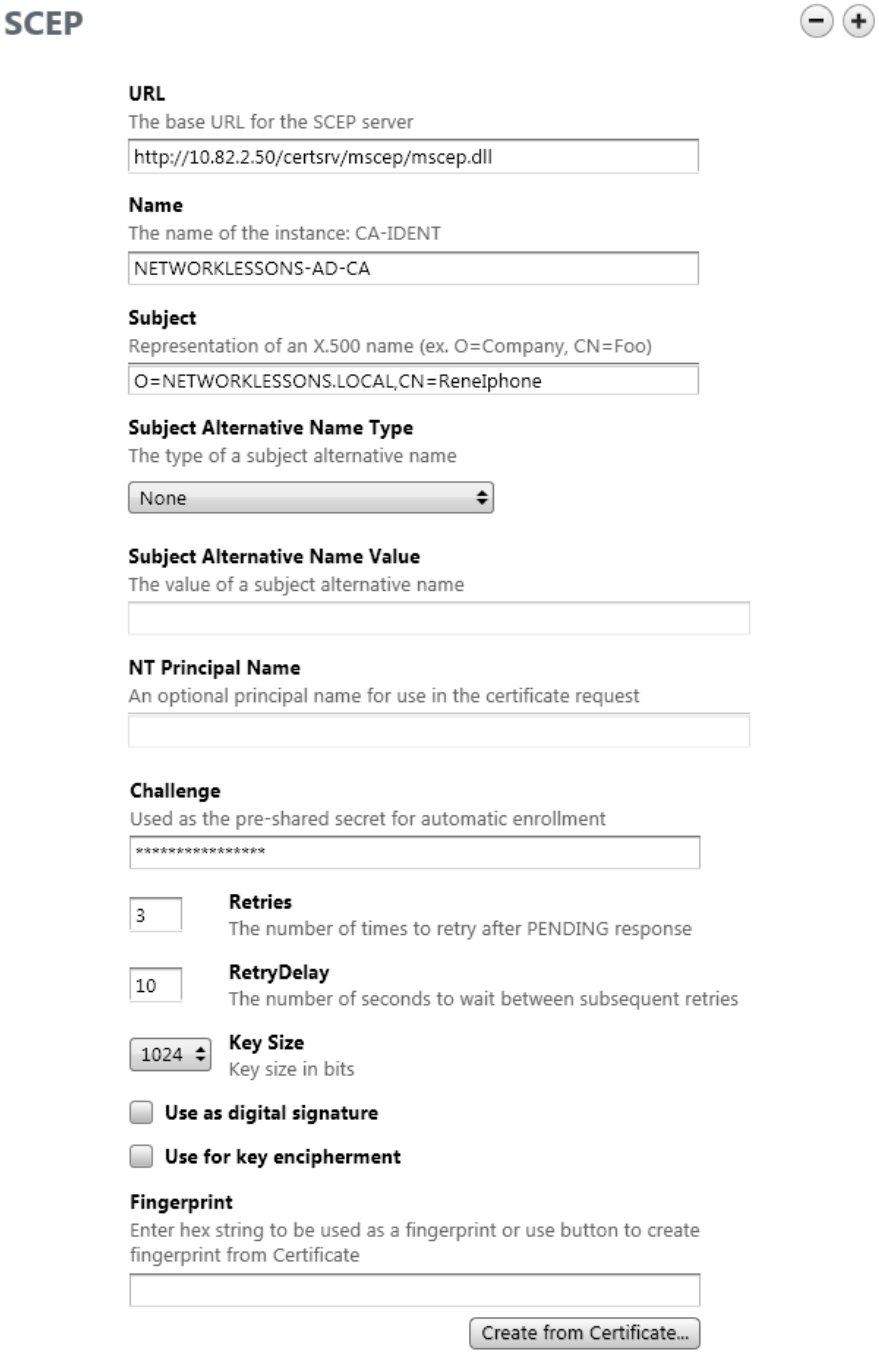

Figura 3.8: iPhone Configuration Utility - SCEP

En este caso, la configuración del protocolo es externa al dispositivo (aprovisionamiento manual del challengePassword y del certificado de la CA) aunque su funcionamiento es interno al dispositivo (no hay funcionalidad proxy).

3.9. Cliente SCEP: Android

Android no dispone de un cliente nativo de SCEP, ni ningún otro mecanismo o protocolo de gestión automática de certificados.

Se puede encontrar en Google Play Store la versión de la aplicación *Mobile@Work* de *MobileIron*, que proporciona alguna funcionalidad más aparte del protocolo SCEP (plataformado de dispositivos, securización de contenidos, control remoto del dispositivo, etc.). Es necesaria la versión servidor del software *MobileIron* (Figura 3.9).

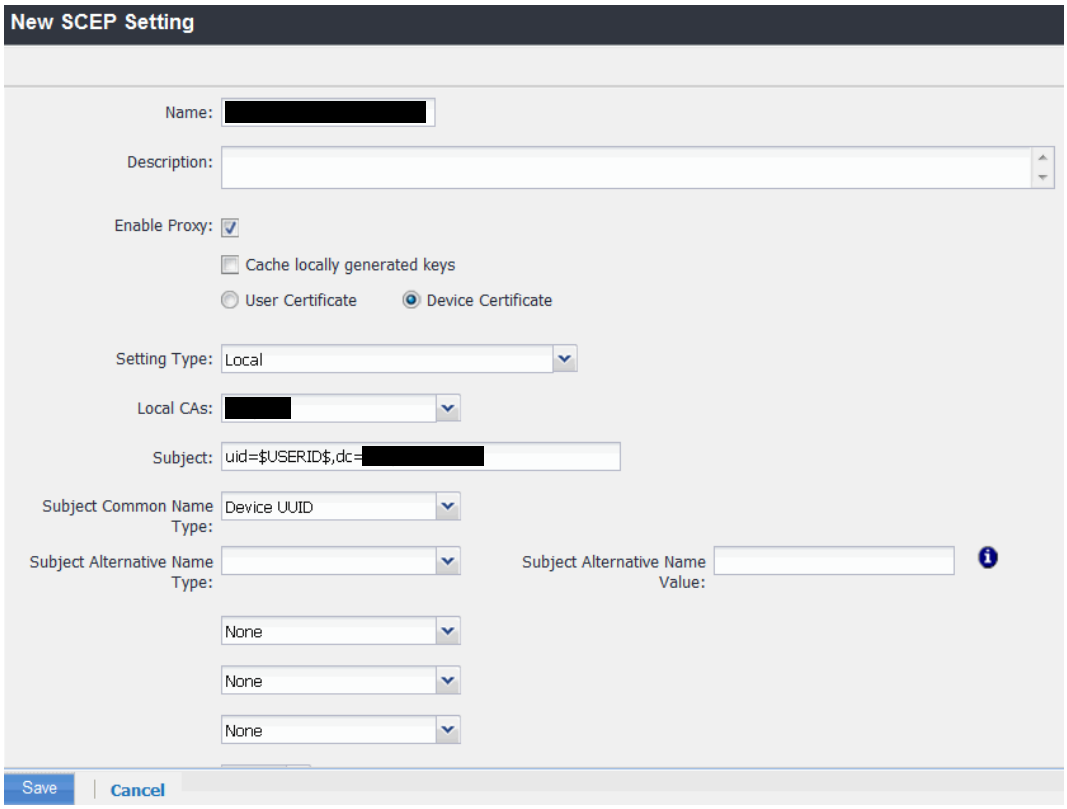

Figura 3.9: Interfaz de servidor *MobileIron*

Capítulo 4

Análisis del diseño

El objetivo principal del trabajo es desarrollar un par de aplicaciones que ejemplifiquen las bondades del protocolo SCEP para su uso, como se ha dicho anteriormente, en dispositivos cliente dentro de una PKI existente en el ámbito e instalaciones del cliente solicitante.

La base del diseño y desarrollo software posterior está compuesta por el estudio y conclusiones obtenidos en el principio del presente documento.

En primer lugar, se especifican las preferencias impuestas por el cliente:

1. Utilización del protocolo SCEP. Aunque se ha comprobado que hay protocolos que incluyen mayor seguridad, el cliente ha probado con éxito diferentes implementaciones comerciales de versiones antiguas del protocolo SCEP, con lo que ha comprobado que funciona con respecto a su PKI instalada.

    A. Las soluciones software de PKI empleadas en el cliente son de Microsoft y de SafeLayer (ya implementan la funcionalidad de servidor SCEP).

2. Implementar la versión 23 del *draft* del protocolo SCEP (última versión publicada).

3. Implementar el cliente móvil para sistema operativo Android, ya que otros sistemas operativos como iOS o BlackBerryOS ya incluyen un cliente SCEP de serie.

4. En principio, se manejarán certificados de autenticación únicamente.

Estas decisiones están fundamentadas en varios aspectos:

- Previo a la realización del proyecto, el cliente contrató una licencia de prueba de un software llamado MobileIron, que implementa un cliente SCEP. Este software ha sido utilizado en pruebas durante un tiempo con resultado satisfactorio, aunque al ser un producto cerrado; era poco personalizable.

  Por tanto, la empresa cliente tiene como preferencia la utilización de dicho protocolo, probado con éxito en la PKI que posee instalada; y que además, tiene implementada su funcionalidad en la solución software que existe actualmente en funcionamiento (desarrollada principalmente por Microsoft y SafeLayer). A pesar de ser un protocolo cuyo *draft* ha expirado y que presenta evidentes deficiencias:

  - Se trata de un protocolo de implementación relativamente sencilla.
  - Como el cliente ha podido corroborar, cumple con todas sus expectativas.
  - El cliente dispone de experiencia previa de uso.

  Se buscaba un protocolo simple debido a que se prevé que el número de dispositivos móviles que harán uso del protocolo será moderadamente elevado (superior a 100 dispositivos), y se teme que una elección errónea haga impracticable el sistema. Además, ninguno de los demás protocolos consigue solucionar de manera eficiente o completa los problemas intrínsecos que posee el protocolo SCEP.

- Para ejemplificar la aplicación del protocolo SCEP:

  - Se decidió la implementación de un cliente orientado a su uso en el sistema operativo Windows para ejemplificar su uso en portátiles, ordenadores de sobremesa, o servidores.
  - Se decidió la implementación de un cliente orientado a su uso en el sistema operativo Android (a partir de su versión 4.1) para ejemplificar su uso en tablets o teléfonos móviles con dicho sistema operativo.
    - Se pretende, en una primera versión, la realización de una aplicación autónoma que inmiscuya al usuario lo menor posible (a diferencia de las alternativas de iOS y BB).

- Se adoptó Java como lenguaje de programación debido principalmente:

  - A su adaptabilidad tanto al sistema operativo Windows (que posee un JDK ampliamente desarrollado) como al sistema operativo Android (que posee un SDK propio en lenguaje Java).

  - A la utilización de la librería jSCEP (desarrollada en Java) y que proporciona las estructuras de datos utilizadas en el intercambio de mensajes del protocolo SCEP haciendo uso de la librería *BouncyCastle*, actualizada a la última versión del *draft*, en contínuo desarrollo, y compatible con las implementaciones de NDES de las últimas versiones de Windows (2008 R2 Server y 2012).
  - A su curva de aprendizaje, y al conocimiento que el desarrollador poseía del lenguaje en cuestión.
  - A su carácter multiplataforma.

Aunque el objetivo último es el desarrollo de la parte servidora del protocolo SCEP dentro de la RA a desarrollar, en una primera fase se busca el desarrollo de un cliente personalizable que implemente el último *draft* del protocolo, como forma de aprendizaje del mismo y como forma de prueba respecto a la instalación existente (XRA KeyOne de la empresa SafeLayer y Microsoft Windows Server 2008).

Durante la especificación de las diferentes alternativas de desarrollo software, se procuró que los aspectos más relevantes a la hora del diseño de la solución software fueran:

- Uso de estándares: se han tenido en cuenta siempre las definiciones de los diferentes estándares involucrados en la definición del protocolo SCEP.
- Uso de Orientación a Objetos: se ha optado por utilizar el paradigma y patrones de diseño de Orientación a Objetos ya que se proporciona una visión del problema muy similar a la realidad, así como un diseño modular al que se le pueden añadir funcionalidades fácilmente.
- Intentar hacer una única aplicación que detecte el sistema operativo en el que se ejecuta, y que según éste, realice el mismo trabajo según se especifique.
  - Dificultad a la hora de compilar un programa Java único orientado a diferentes máquinas virtuales.

Se valoró el desarrollo de la aplicación como un servicio Web, pero ante la dificultad del mismo, se optó por el desarrollo de una aplicación núcleo a la que, según el sistema operativo se le aplican los módulos de lógica correspondientes (2 aplicaciones diferentes).

Capítulo 5

Diseño e implementación

Aunque el lenguaje de programación es multiplataforma, se ha tenido que desarrollar dos versiones del programa debido a las restricciones que se imponen desde la implementación de la Máquina Virtual de Java que se realiza para Android.

Durante la búsqueda de información del protocolo SCEP, se encontró una librería open-source llamada jSCEP, que proporciona la funcionalidad de conexión y de envoltura de los mensajes que se envían al servidor SCEP. Esta librería es la base de la alternativa comercial MobileIron.

Durante la evaluación de la librería, se encontraron algunas deficiencias en su diseño que tuvieron que ser adaptadas (principalmente a la hora del desarrollo en Android) y rediseñadas (por ejemplo, ausencia de pasarela SSL para conectar al servidor por medio de HTTPS).

En cualquier caso, suponía una ventaja enorme disponer de dicha librería para adaptarla al producto. Una vez comprobado que su licencia permitía su modificación y utilización, se incluyó como requisito de diseño.

Las modificaciones realizadas a la librería y las razones de su realización fueron:

- Cambio de la librería *BouncyCastle* por *SpongyCastle*.
    - *SpongyCastle* es un fork de *BouncyCastle* que permite la ejecución de todas las funciones criptográficas del *BouncyCastle* original en Android, por lo que, no suponía un cambio en la forma de programación y ampliaba su uso al sistema operativo Android.
- Añadido de método de obtención automática de *challengePassword*.

- Uno de los requisitos del cliente consistía en garantizar la mínima interacción del usuario y el menor número de trámites posibles en la inscripción del dispositivo en la PKI. Como las pruebas del cliente iban a estar involucradas sobre una PKI basada en Windows Server, se aprovechó la posibilidad de obtención del *challengePassword* mediante un API que la obtuviera por medio de la aplicación `mscep-admin` (implementada como parte del servicio NDES de Windows Server).
    - Esto provocaba que el cliente tuviera que tener precargado un usuario y contraseña de administrador de la CA SCEP, para poder acceder a la aplicación de generación de *challengePassword* (que requería de ejecución privilegiada).
        - Argumento principal de la necesidad de securización de las propiedades del programa.

- Añadido de método de conexión HTTPS [W7] con servidor SCEP
    - Por defecto, la librería sólo admitía la realización del envío de mensajes SCEP por medio del protocolo HTTP. Sin embargo, y para añadir seguridad extra (ya que la definición del protocolo lo permite), se añadieron las primitivas y soporte para la conexión mediante HTTPS con el servidor SCEP.
- Resolución de pequeños problemas de compatibilidad con la implementación de SCEP de Microsoft y SafeLayer (principalmente el tratamiento del campo `MESSAGE` en los mensajes SCEP).

5.1. Librerías utilizadas

5.1.1. Java SCEP: jSCEP

Java SCEP o jSCEP es una librería desarrollada en Java que implementa la última versión del *draft* del protocolo SCEP. Es una librería de carácter libre o *open-source* que proporciona los mecanismos y estructuras de datos básicas para la realización de un cliente o servidor SCEP.

La licencia de distribución de dicha librería es Creative-Commons-3.0 BY (*Share-Adapt free with Attribution*).

Esta librería implementa la última versión del *draft* publicada del protocolo SCEP, y hace uso de la librería criptográfica *BouncyCastle* [W14].

Los diagramas UML especificados en las figuras 5.2 y 5.1 no aparecen completos debido a la complejidad de los mismos. Además, han sido divididos debido a que sus dimensiones son superiores al área legible del documento.

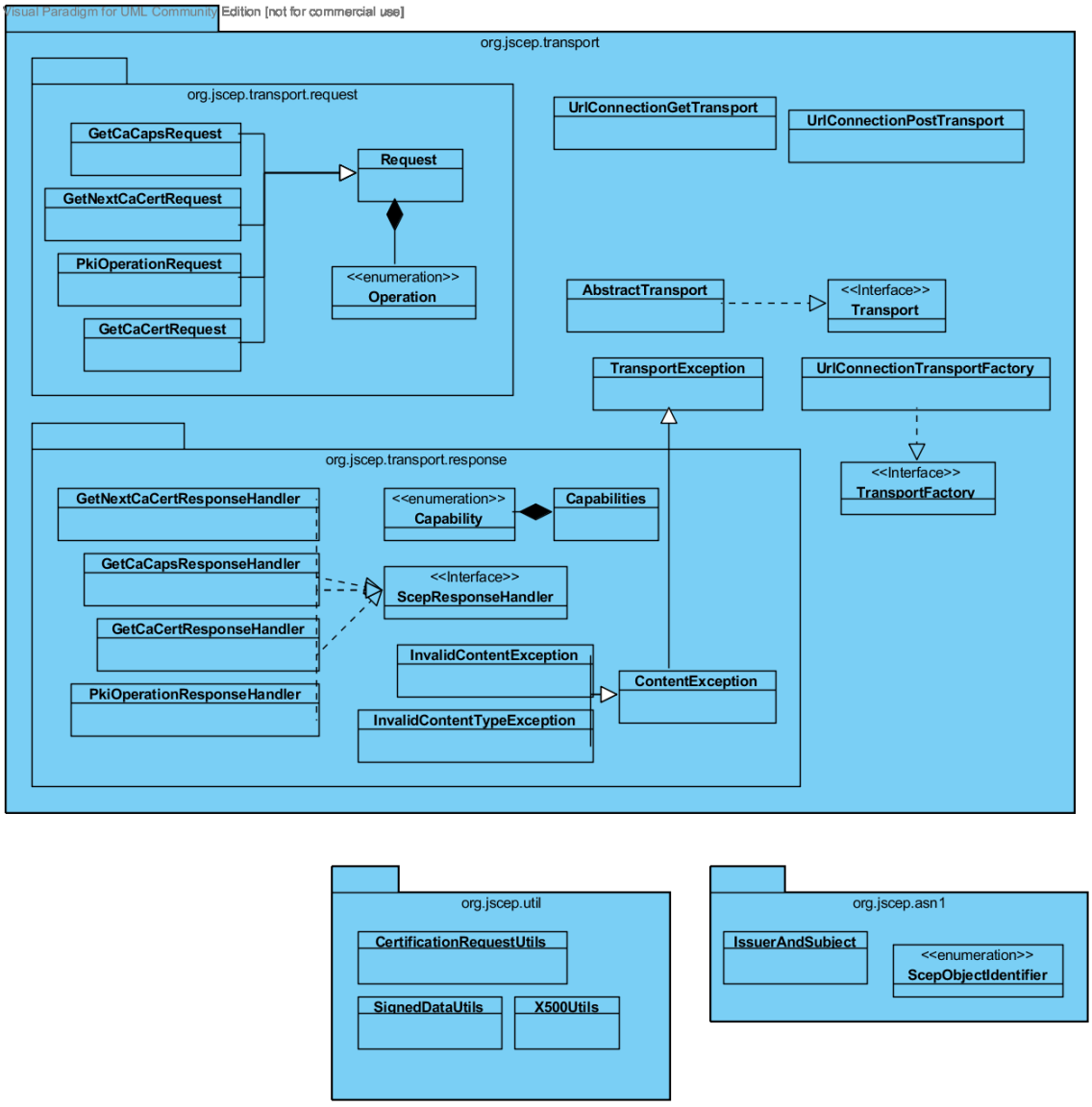

Figura 5.1: jSCEP - Diagrama UML (Parte 1)

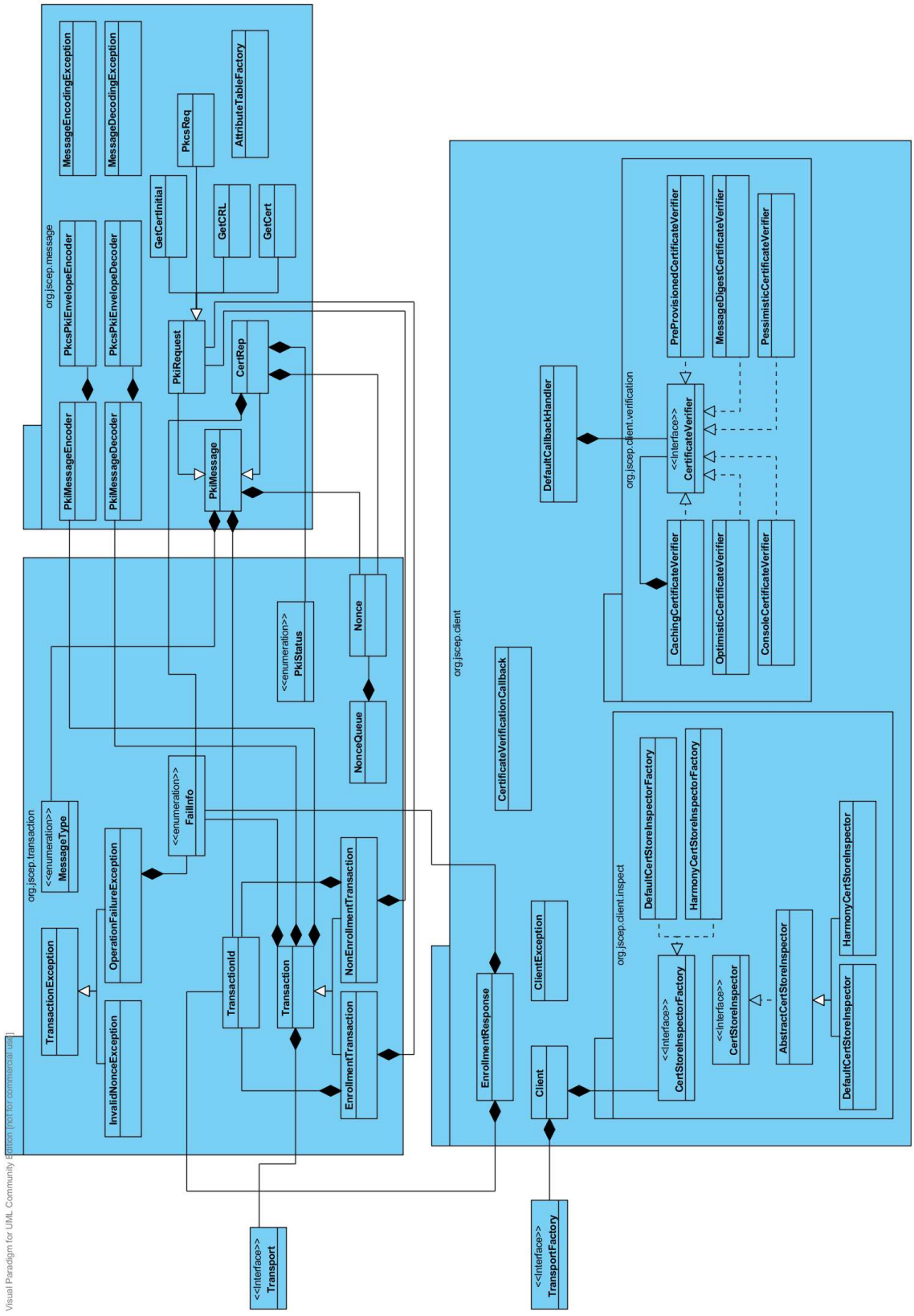

Figura 5.2: jSCEP - Diagrama UML (Parte 2)

A continuación se detallan las 2 clases más representativas e importantes de la librería: `org.jscep.client.Client` y `org.jscep.transaction.Transaction`.

org.jscep.client.Client (Figura 5.3)

La clase `org.jscep.client.Client` proporciona el API de acceso al envío y recepción de los diferentes mensajes implementados en el protocolo.

Los métodos públicos tienen dos formas de invocación: incluyendo el `String profile` o no. Esto es debido a una característica ya comentada en el apartado 3.4.1, por la cual, dependiendo de la implementación del servidor SCEP, puede o debe requerirse el envío de un atributo `message` (con el valor adecuado: id de dispositivo, id de CA, etc.), en el envío de los mensajes **GetCACert** o **GetCACaps**.

Figura 5.3: jSCEP - Clase *Client*

org.jscep.transaction.Transaction (Figura 5.4)

La clase `org.jscep.transaction.Transaction` proporciona la estructura de los datos que se ven envueltos en una transacción SCEP:

- `encoder`: codificador de envío (construido a partir de la clave pública de la CA y la privada del dispositivo).
- `decoder`: decodificador de recepción (construido a partir de la clave privada del dispositivo).
- `transport`: protocolo de transporte (HTTP/HTTPS)
- `state`: estado [SUCCESS, PENDING, REJECTED]
- `certStore`: almacén de certificados envuelto en la transacción (respuesta de **GetCACert**, **GetNextCACert**, etc.

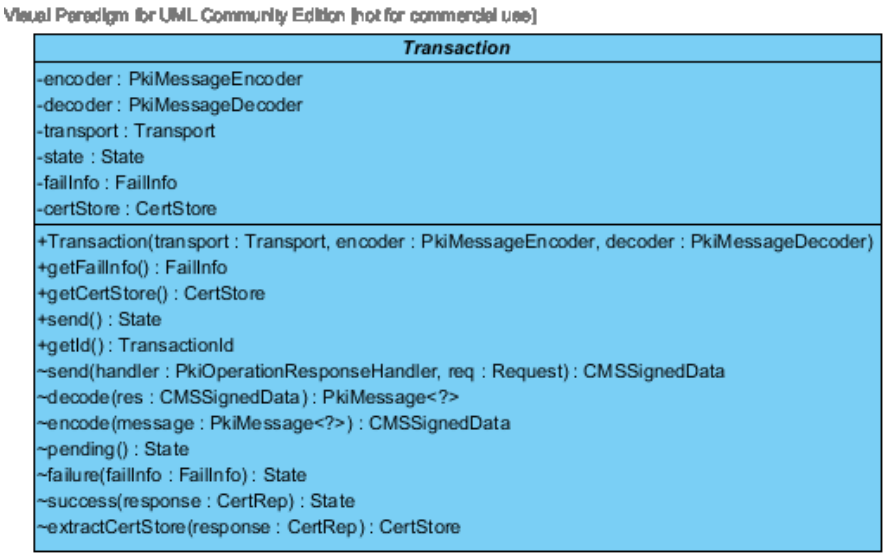

Figura 5.4: jSCEP - Clase *Transaction*

Cambios realizados

Como se ha comentado anteriormente, se ha añadido el soporte al protocolo de transporte HTTPS. Esto no resuelve de manera determinante la seguridad del

protocolo, pero garantiza que el intercambio de información se realiza cifrado por un medio no seguro.

De igual manera, se ha adaptado la librería al entorno de pruebas. Por defecto, la librería asegura que su funcionamiento con la implementación del servidor SCEP que dispone Windows Server 2008 es completa. La realidad mostró que no es exactamente así (Ver apartado 3.6). Por ello, se implementó un verificador del certificado de la CA obtenido mediante **GetCACert** que utilizara el algoritmo MD5 (desfasado, pero es el que proporciona Windows Server 2008 para realizar la validación del certificado).

También se añadió el tratamiento para que no resultara una excepción o comportamiento anómalo cuando el servidor SCEP de Windows Server 2008 no dispusiera del mensaje **GetCACaps** o **GetNextCACert**. El comportamiento añadido se corresponde al tratamiento mínimo estricto impuesto por el *draft*.

5.1.2. Otras librerías

Otras librerías utilizadas en el desarrollo del proyecto:

- Apache Commons
  - Apache Commons - Codec 1.8
  - Apache Commons - IO 2.4
  - Apache Commons - Lang 2.6
- Simple Logger Factory 4 Java (Slf4j)
- JTelegraph
- JCarrier
- SpongyCastle 1.49
- Librería de securización de configuración (propietaria).
- Android Support v.4

Estas librerías han sido utilizadas para diferentes funcionalidades dentro del programa (cálculo de Base64, escritura de logs, notificaciones de escritorio, funcionalidad criptográfica, etc.)

5.2. Servlet de generación RSA

Para proporcionar la funcionalidad de generación externa (al dispositivo) de pares de claves por medio del algoritmo RSA, se ha implementado un servlet en Java que, por medio de peticiones HTTP POST y especificándole una serie de parámetros; genera y devuelve el par de claves por medio de HTTPS [W16].

Inicialmente, ha de disponerse del certificado de la CA emisora del servlet (en este caso, al estar en la misma máquina que la CA, coincide con el certificado que se puede obtener por medio del mensaje SCEP **getCACert**) para poder realizar la autenticación del servidor (por cuestión de seguridad).

Seguidamente, el conector Java que realiza la conexión al servlet, genera una clave simétrica. Esta clave simétrica es la clave que usará el servlet para cifrar el par de claves mediante el algoritmo TripleDES de BouncyCastle [W15].

Esa clave simétrica es enviada al servlet encriptada con la clave pública del certificado validado del servidor, junto con los parámetros de generación de clave.

El servlet realiza la generación de claves mediante algoritmo RSA por medio del proveedor de servicios criptográficos de la librería *BouncyCastle* [W14].

5.3. Aplicación Android

La aplicación Android desarrollada se compone de varios componentes principales:

- Un Activity principal que hace las funciones de entrada a la aplicación cuando se hace click en el icono de la aplicación instalada.
- Un IntentService (servicio en segundo plano) que realiza las comprobaciones que implica la lógica de manejo automático de los certificados involucrados en el intercambio definido en el protocolo SCEP.
- Un BroadcastReceiver que realiza las funciones de punto de entrada a eventos internos del dispositivo (alarmas programadas o arranque del sistema) y realiza la acción determinada.
- Un NotificationManager que se encarga de mostrar las notificaciones al usuario por pantalla.

La aplicación requiere de los siguientes permisos de ejecución en la instalación (Figura 5.5):

Figura 5.5: Permisos instalación aplicación Android

Estos se justifican por el acceso al almacén de certificados de Android, por el acceso al servidor SCEP y por el arranque al inicio del sistema operativo de la comprobación rutinaria de certificados.

Su diagrama de clases simplificado es el siguiente:

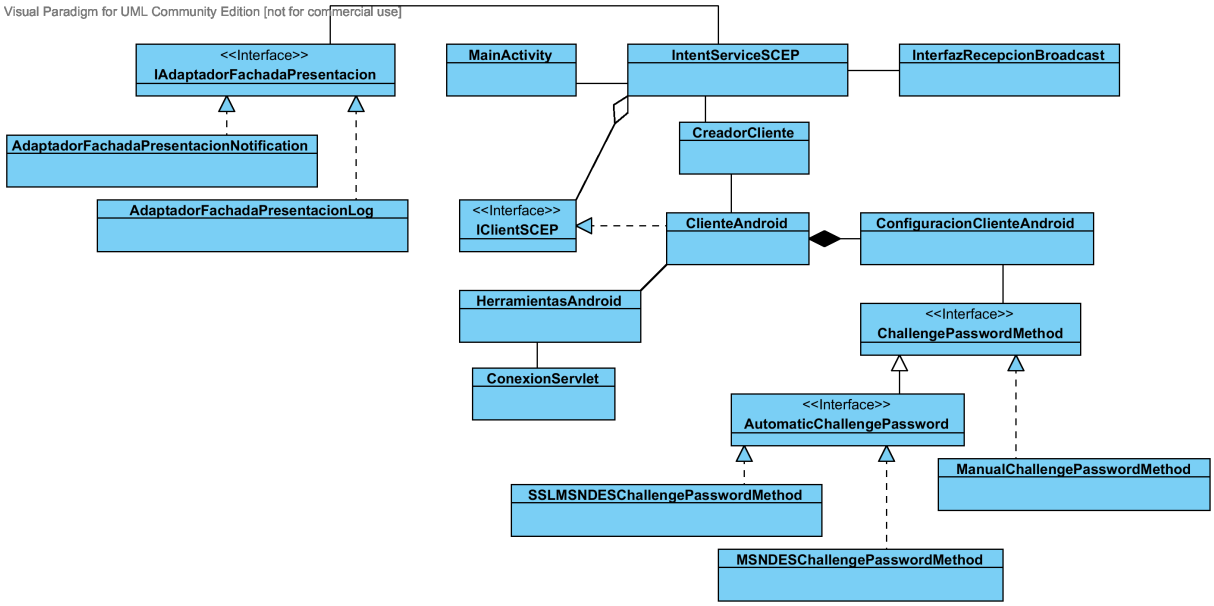

Figura 5.6: Diagrama de clases: Android

Las clases principales del programa son:

- `InterfazRecepcionBroadcast`: Clase que permite la iniciación del programa por medio de la recepción de las alarmas registradas en el sistema operativo de programación de las tareas periódicas (reinstalación, comprobación, renovación, etc).
- `IntentServiceSCEP`: es el corazón del programa. Encargado de la lógica de la comprobación periódica de los certificados y de su instalación. Ordena la creación del cliente SCEP. Envía las notificaciones al `AdaptadorFachadaPresentacion` según se ordene en la invocación del programa.
- `ClienteAndroid`: contiene la lógica de las operaciones del cliente SCEP. Coordina el flujo de la lógica de las operaciones con las clases `ConfiguracionClienteAndroid` y `HerramientasAndroid`
- `ConfiguracionClienteAndroid`: carga la configuración externa del programa y proporciona su interfaz de acceso para la creación y lógica del cliente.
- `HerramientasAndroid`: proporciona alguna operación criptográfica como firma, creación de CSR, o generación de claves RSA.

Internamente, se guarda una copia de los certificados residentes en el dispositivo (tanto los autofirmados que se crean para las solicitudes, como los recibidos del servidor SCEP), para poder hacer la comprobación de los instalados en el almacén central de Android.

La razón por la que se ha optado hacer esto es porque aunque sí son accesibles las claves y los certificados instalados por la aplicación en el almacén de Android, supondría más mensajes de interfaz de usuario que implicarían la acción de éste. De esta forma, se reducen los mensajes y la interacción del usuario y no afecta al funcionamiento ni de la aplicación ni del protocolo SCEP.

5.4. Aplicación Windows

La aplicación Windows desarrollada se compone de varios componentes principales:

- Un Hilo principal que se encarga de realizar las comprobaciones periódicas. Éste permanecerá en estado suspendido mientras no esté en funcionamiento
- Un Pool de hilos que ejecutará las tareas programadas (un hilo por tarea) y en exclusión mutua para garantizar el acceso al recurso compartido (en este caso, el almacén central de Windows).

Se hace acceso al almacén central de certificados por medio del proveedor de servicios criptográficos `SunMSCAPI`, instalado en la JRE de Java.

Su diagrama de clases simplificado es el especificado en la figura 5.7.

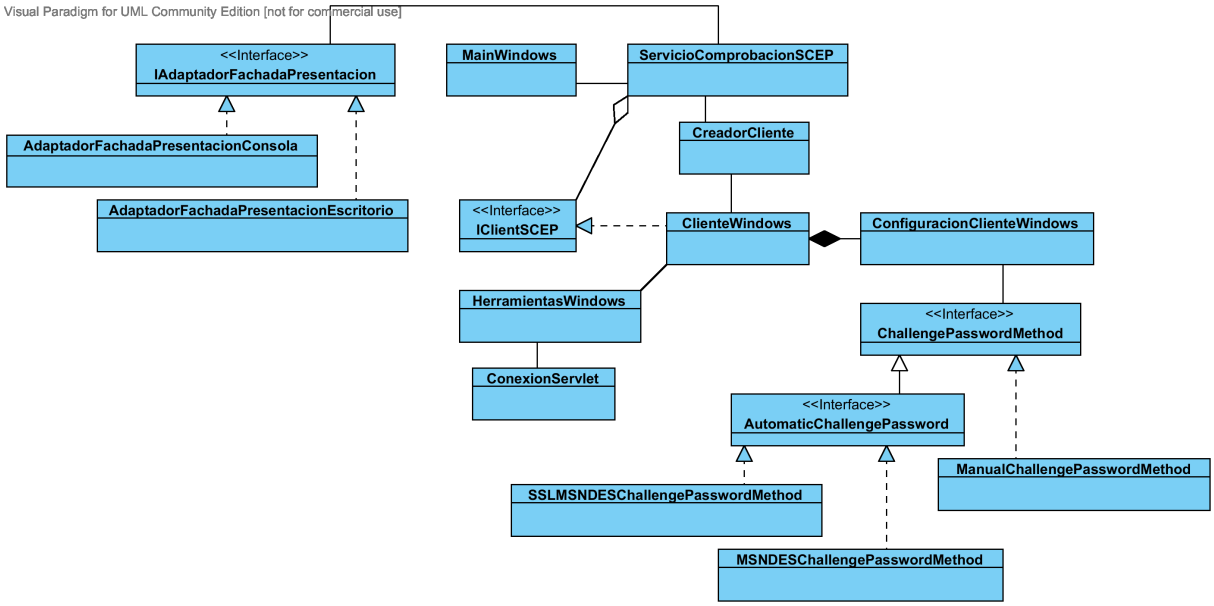

Figura 5.7: Diagrama de clases: Windows

El único punto de entrada al programa es la clase MainWindows, que se encarga de ordenar la creación del ServicioComprobacionSCEP.

Las clases principales del programa son:

- `ServicioComprobacionSCEP`: es el corazón del programa. Encargado de la programación de las tareas de comprobación, renovación y reinstalación. Ordena la creación del cliente SCEP y de la instalación de los certificados. Envía las notificaciones al `AdaptadorFachadaPre- sentacion` según se ordene en la invocación del programa.

- `ClienteWindows`: contiene la lógica de las operaciones del cliente SCEP. Coordina el flujo de la lógica de las operaciones con las clases `ConfiguracionClienteWindows` y `HerramientasWindows`

- `ConfiguracionClienteWindows`: carga la configuración externa del programa y proporciona su interfaz de acceso para la creación y lógica del cliente.

- `HerramientasWindows`: proporciona alguna operación criptográfica como firma, creación de CSR, o generación de claves RSA.

5.5. Fichero de configuración

Para hacer personalizable (en cuanto a la configuración de parámetros) el programa, y haciendo caso a las recomendaciones del cliente; se incluye un archivo de configuración del programa encriptado con una librería propietaria de la empresa desarrolladora para evitar la obtención de información susceptible de causar un problema de seguridad importante (como por ejemplo el hash del certificado de la CA, o la URL de acceso al sistema de generación de challengePassword).

Este fichero de configuración se proporciona en la instalación del software, con lo que, los datos del dispositivo deberían de haber sido proporcionados

A continuación se muestran algunos de los atributos más representativos del fichero de configuración:

- Identificador del dispositivo.
- Datos del servidor SCEP.
  - URL del servidor SCEP.
  - Hash del certificado de CA.
  - Obtención del challengePassword automáticamente (NDES) o challengePassword preprovisionado.
- Datos del cliente.
  - Identificación del dispositivo.
  - Número de serie del último certificado emitido.
- Datos de configuración.
  - Ruta del KeyStore.
  - Parámetros de conexión: número de reintentos, período de comprobación del estado de los certificados, etc.

Como parece evidente, el fichero contiene información sensible de la PKI, accesible solamente al programa por medio de la librería de securización de propiedades proporcionada por la empresa desarrolladora. Esta es de uso interno, validada mediante su utilización en numerosos proyectos de ámbito público y privado, y que aplica métodos criptográficos avanzados.

Capítulo 6

Pruebas y resultados

El entorno de pruebas en el que se han probado las soluciones software son:

- Entorno Windows
  - Máquina de pruebas: HP EliteBook 8470p con Microsoft Windows 7 Enterprise
  - Servidor SCEP: Microsoft Windows Server 2008 R2.
    - Funcionando sobre máquina virtual en Oracle VirtualBox (versión 4.3.6).
  - HotSpot Wifi Virtual: HP Connection Manager / Microsoft Virtual WiFi Miniport Adapter.
  - Servidores Web:
    - Microsoft IIS 6: funcionalidad SCEP
    - Apache Tomcat 6: generación externa de claves, y prueba de autenticación de cliente.
- Entorno Android
  - Máquina de pruebas: BQ Aquaris 4.5 con Android 4.1.1
  - Máquina virtual de pruebas: Oracle VirtualBox (versión 4.3.6) con Android 4.3 (versión x86).

6.1. Servlet de autenticación de cliente SSL

Para la prueba final del uso de los certificados obtenidos en los dispositivos, se ha implementado un Servlet en Java como controlador de una aplicación JSP (*Java Server Page*) [W15, W16].

Este servlet se encarga de recibir el certificado que presenta el cliente para autenticarse por SSL, realizar una serie de comprobaciones, y mostrar como respuesta un resumen de las propiedades del certificado.

Las comprobaciones a realizar sobre el certificado presentado:

- Verificación de la entidad emisora: se comprueba si el certificado ha sido firmado por una CA admitida.
  - En este caso, la comprobación se realiza sobre una CA raíz.
- Verificación del estado de revocación del certificado.
- Verificación de la fecha de caducidad del certificado.

Al introducir la dirección del servlet de autenticación, se inician las comprobaciones asociadas al protocolo HTTPS. El navegador comprobará el certificado que presenta el servidor con el certificado de la CA que tiene instalado.

Si la comprobación es correcta, se buscarán los certificados de cliente instalados en el almacén que hayan sido emitidos por la CA que se presenta. Si no, se pedirá al usuario la aceptación del certificado de servidor advirtiéndole de que no ha podido ser comprobado con respecto a un certificado de una entidad de confianza.

Dependiendo del sistema y del navegador utilizados, se muestran los siguientes diálogos de selección de certificado de cliente (Figuras 6.1, 6.2 y 6.3):

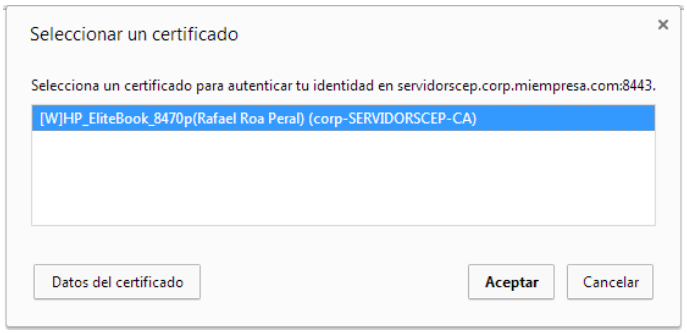

Figura 6.1: Autenticación SSL Windows - Google Chrome

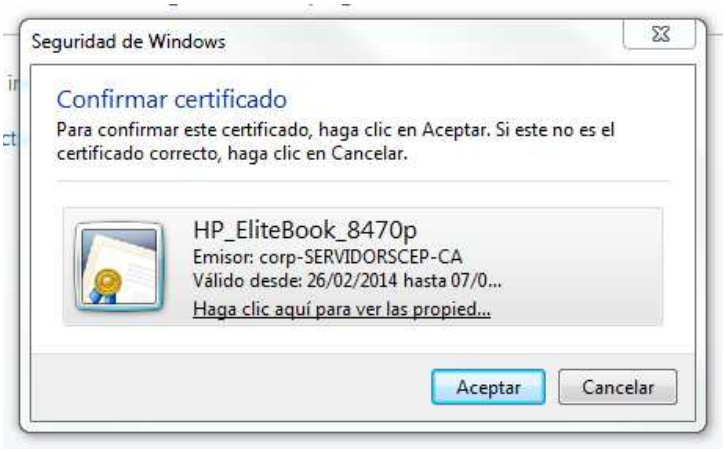

Figura 6.2: Autenticación SSL Windows - Internet Explorer

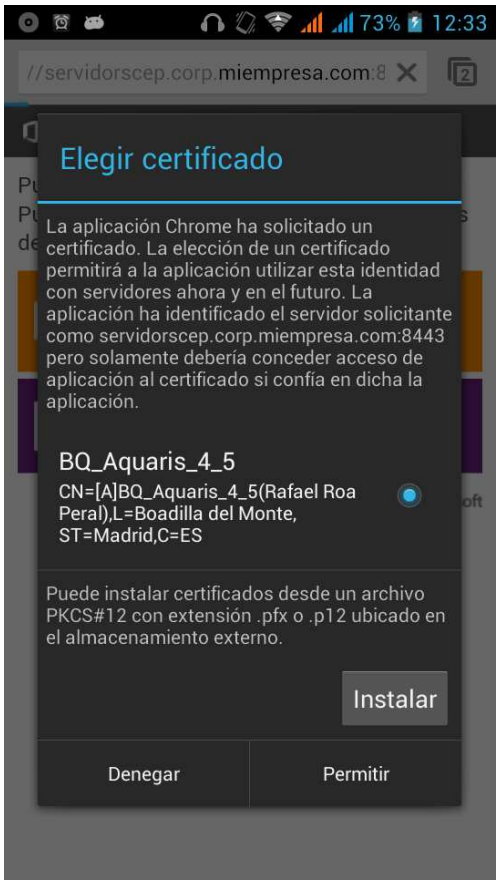

Figura 6.3: Autenticación SSL Android - Google Chrome

La aplicación recoge el certificado que se ha presentado como certificado de autenticación de cliente y realiza las comprobaciones anteriormente citadas. Los posibles resultados son:

- Certificado de cliente caducado (Figura 6.4).
- Certificado de cliente revocado (Figura 6.5).
- Certificado de cliente válido (Figura 6.6).

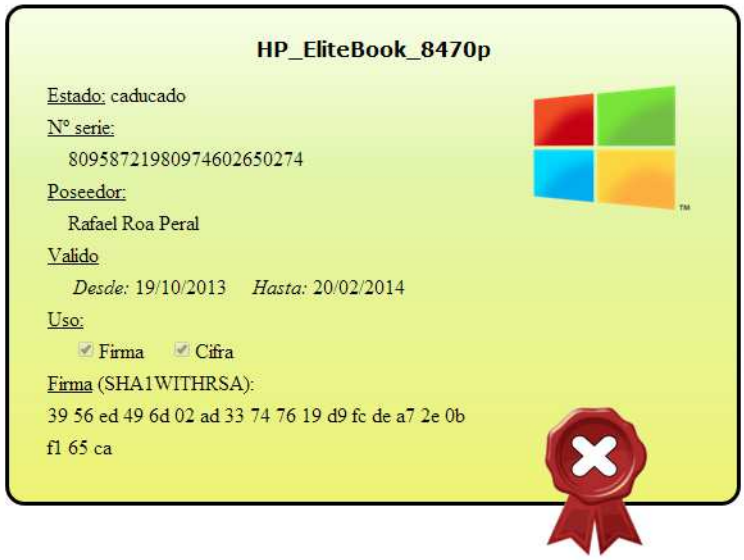

Figura 6.4: Servlet Autenticación - Certificado caducado Windows

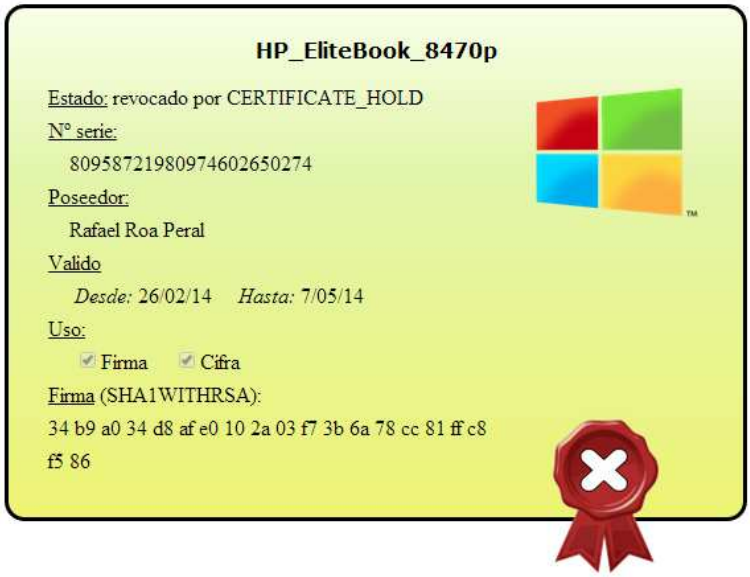

Figura 6.5: Servlet Autenticación - Certificado revocado Windows

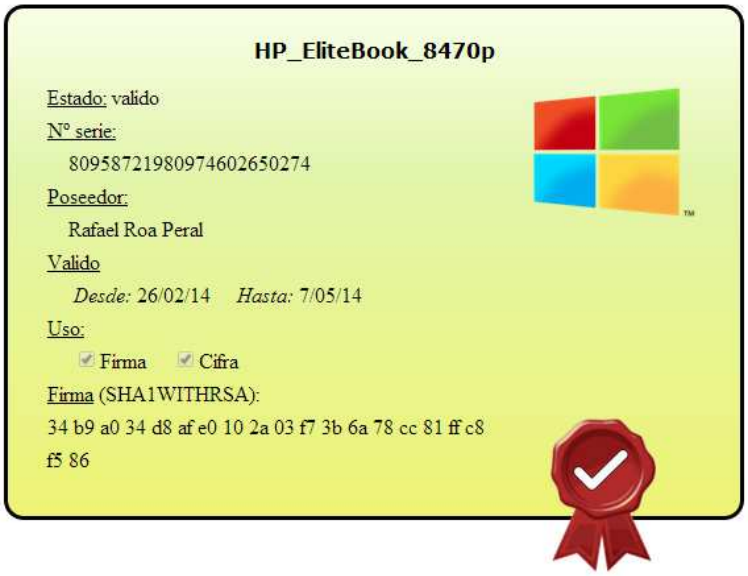

Figura 6.6: Servlet Autenticación - Certificado válido Windows

Si el certificado es emitido para un dispositivo Android, los resultados son equivalentes; aunque se presentan con algunos cambios, como se muestra en la figura 6.7.

Figura 6.7: Servlet Autenticación - Certificado revocado Android

Capítulo 7

Conclusiones y líneas futuras

7.1. Tiempo empleado

Las distintas tareas realizadas durante la ejecución del PFC han sido agrupadas de la siguiente forma:

1. Fase de aprendizaje: en la que se ha agrupado cualquier tarea de adquisición de conocimiento enmarcada en el PFC: conceptos básicos de seguridad, panorámica de protocolos de gestión de certificados, protocolo SCEP, administración básica de una CA, programación orientada a Android o redacción de documentos en LaTeX.

2. Diseño y especificación de requisitos: en la que se enmarcan las entrevistas con el cliente, así como la realización de los diagramas y formularios correspondientes: alternativas de diseño, interfaces, alto nivel, etc.

3. Implementación: realización en código del diseño planificado de todos las aplicaciones y productos derivados del proyecto.

4. Pruebas: planificación y realización de pruebas de funcionamiento de los productos software.

5. Documentación: realización del presente documento, así como la documentación interna del desarrollo software realizado.

El reparto del tiempo empleado en la realización del proyecto se muestra en el gráfico de sectores siguiente (Figura 7.1).

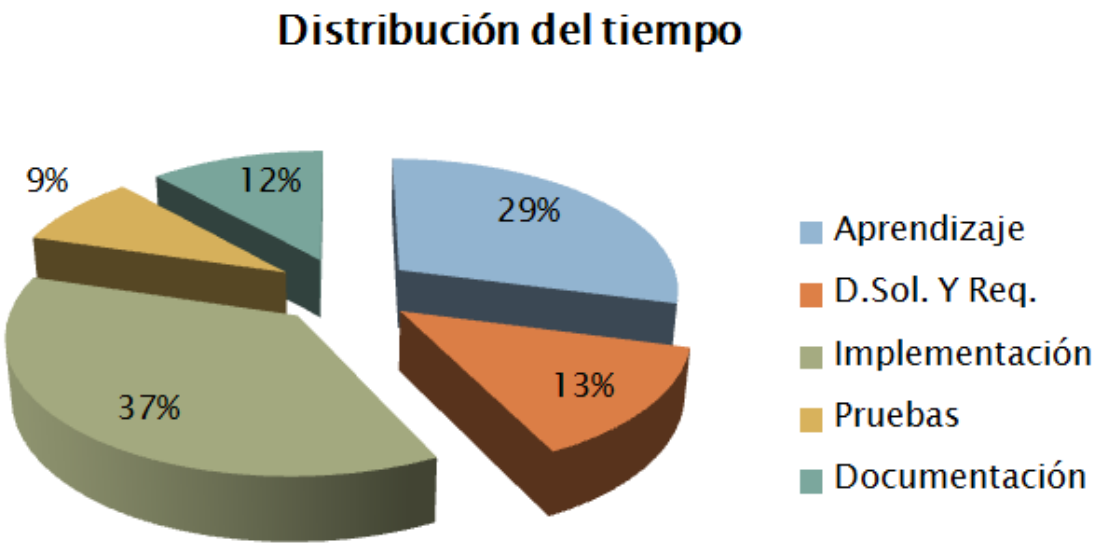

Figura 7.1: División del tiempo empleado

7.2. Coste asociado

A continuación se muestra un pequeño análisis del tiempo empleado en la realización del proyecto, así como la presentación del coste asociado.

En este trabajo sólo ha habido una persona dedicada a la realización de todas las tareas presentadas. Esta persona ha estado trabajando el el proyecto durante 5 meses, a razón de 7 horas diarias. Debido a su condición de becario, el salario por hora trabajada era de 4,5 €/hora (700 €mensuales).

Adicionalmente, se le proporcionó un ordenador portátil para la realización del trabajo (cuyo coste debe ser añadido al coste del proyecto), valorado en 1000 €.

Las licencias software utilizadas han sido versiones de prueba o versiones de uso académico, por lo que no hay que añadir ningún coste adicional en términos de licencias software.

Para el cálculo final de coste de mano de obra, se ha tomado una media de 22 días laborables por mes de trabajo, siendo el coste final asociado al proyecto:

$$coste\_ordenador + (4,5\ €/\text{hora} * 7\ \text{horas/dia} * 22\ \text{dias/mes} * 5\ \text{meses}) = 4465\ €$$

7.3. Líneas futuras

En cuanto a las líneas futuras derivadas del desarrollo del PFC, así como posibles mejoras a implementar, se destaca en primer lugar que se trata de un producto empresarial. Esto quiere decir que va a ser continuado (tal y como se ha resaltado en diferentes lugares del presente documento) y ampliado con la versión servidora del protocolo SCEP, mediante su inclusión en la funcionalidad de una RA de próximo desarrollo.

Se han dividido las diferentes líneas futuras en: líneas futuras aplicables al propio protocolo SCEP, y las aplicables al software desarrollado (Windows y Android).

7.3.1. Protocolo SCEP

En cuanto a las líneas futuras aplicables al protocolo SCEP, existen una serie de problemas atribuibles a las vulnerabilidades conocidas del protocolo.

Por ejemplo, existe una vulnerabilidad por la cual un atacante, si se hace con un challenge password, y si el servidor no requiere autenticación alguna, puede solicitar la emisión de un certificado digital en nombre de otro; lo cual le daría permiso para acceder a las aplicaciones de la PKI que hicieran uso de esa plantilla de certificado.

Este problema viene derivado de uno de los inconvenientes que posee el protocolo SCEP, y es la ausencia de una autenticación inicial adicional al uso del *challenge password*[W21]. Por ello, y aunque signifique abandonar la referencia a la documentación del protocolo; podría añadirse algún método de autenticación adicional por medio de algún método, como por ejemplo:

- Registro previo del dispositivo y asociación con un challenge password.
  - Esto requiere de funcionalidad adicional al protocolo SCEP y que debe ser implementada por el servidor.
- Introducir parámetros adicionales en la solicitud de certificación, como por ejemplo un usuario y contraseña autorizado en la PKI, u otro proporcionado previamente y asociado a la identidad solicitante del certificado.
- Introducir un modo de "cese de función" del programa por casos especiales (por ejemplo, finalización del contrato laboral del trabajador)
  - Puede ser a nivel de RA o añadiendo la funcionalidad extra al servidor del protocolo SCEP.

Adicionalmente, podría modificarse alguna característica relativa a la plantilla de emisión de certificado que utiliza la CA. En este caso, podría incluirse algún tipo de OID adicional como IMEI, nombre y tipo de dispositivo, etc. (para distinguir el uso o plantilla de certificado).

Como el objetivo último del desarrollo es el desarrollo de la parte servidora del protocolo para obtener una solución software completa, las modificaciones al protocolo quedan a juicio del cliente, debido a las diferentes políticas de seguridad que pueda tener en su PKI.

7.3.2. Software desarrollado

En cuanto al software desarrollado, las posibles líneas futuras están relacionadas con el uso que se le dé al propio software:

- Traducción a diferentes idiomas (Definición de archivos de lenguaje)
- Adelgazar el tamaño del programa (Reducción de librerías, optimización de código)
- Adaptar a diferentes implementaciones de Máquina Virtual de Java (JVM) para su inclusión en sistemas operativos no convencionales (AIX -IBM-, por ejemplo).
- Adaptar a otros entornos y lenguajes de programación (Windows Phone, BlackBerry OS, iOS, .Net, Java WebServices, etc.).
- Adaptar a otros protocolos, como por ejemplo EST o CMP, que son los protocolos más parecidos a SCEP.
- Actualmente sólo soporta la gestión de un único perfil y certificado. Una posible modificación podría ser la adaptación del programa al manejo de diferentes perfiles de configuración para diferentes PKI, o para diferentes certificados dentro de la misma PKI (uno de cifrado, otro de autenticación, firma, etc.)
    - Para que la aplicación soportara la gestión correcta de certificados distintos de sólo autenticación, habría de implementarse lógica adicional (por ejemplo: control de borrado de certificados para evitar perder información previamente cifrada con un certificado revocado).
- Añadir algún módulo de aviso en caso de caducidad de certificado: email (SMTP), notificación SMS, etc.

- Aplicación de criterios adicionales de securización de código Java (como por ejemplo los definidos en el *CERT Oracle Secure Coding Standard for Java*).

En cuanto a la distribución, el software desarrollado está pensado como una prueba de concepto del cliente de protocolo SCEP, por lo que sólo ha sido pensada su instalación en el dispositivo de pruebas. Para permitir su distribución entre los diferentes dispositivos, habría que desarrollar algún sistema de preprovisionamiento de los datos del dispositivo (a semejanza de los vistos en BlackBerry y iOS). Este sistema podría ser un servicio accesible vía Web por el que se obtiene el ejecutable compilado para el dispositivo.

A continuación, se dividirán las líneas futuras según el producto software desarrollado:

7.3.3. Servlet de generación de claves

Este servlet implementado únicamente para realizar una generación externa de claves RSA (ideado para generación rápida de claves superiores a 4096 bits), podría ser ampliado con opciones de firma adicionales (para evitar la firma de contenido en el dispositivo con claves muy grandes).

De igual manera, podría ser la versión inicial de un próximo desarrollo de proxy SCEP o simplemente un proxy de generación de challengePassword (quizás el único aspecto controvertido del protocolo).

7.3.4. Versión escritorio

La versión de escritorio está originalmente pensada como prueba de funcionamiento del protocolo en un ordenador personal de empleado en un entorno de PKI, pero podría adaptarse para actuar como un servidor de gestión de certificados.

Un ejemplo de uso de este sistema sería: un servidor único en conexión con la RA a través del protocolo SCEP (con la evidente mejora en cuanto a la seguridad, ya que sólo habría que securizar la conexión con dicho servidor), y que fuera el encargado de gestionar automáticamente los certificados de varios dispositivos o máquinas que estén a su cargo (routers, otros servidores o máquinas dedicadas, máquinas personales, etc.), y a los que debería proporcionar el certificado bien por un directorio de certificados compartido, o por algún protocolo adicional.

Con esta nueva funcionalidad, no haría falta portar el código a los diferentes sistemas operativos o versiones de la JVM que puedan estar presentes en los diferentes dispositivos que estén conectados al servidor (con el consiguiente ahorro en tiempo y costes).

Además, se podría realizar una conexión ad-hoc entre el servidor y los diferentes dispositivos al cargo (usando parámetros de seguridad más allá de los definidos por el protocolo SCEP).

7.3.5. Versión Android

En primer lugar, lo fundamental es adaptar la aplicación móvil a versiones posteriores de Android (comenzando por la versión 4.3 Jelly Bean, que incluye posibilidad de anulación de certificados por 'lista negra' '[W13]), que incluyen cambios en la gestión de certificados. Esto posibilita que se pueda aumentar la funcionalidad del cliente SCEP de Android.

Otra posible funcionalidad a implementar es la posibilidad de que el cliente SCEP sea capaz de gestionar certificados y perfiles SCEP de varias PKI, para que se pueda hacer la gestión automática de certificados en diferentes PKI que lo tengan disponible; pero usando un único dispositivo [W12].

Esta funcionalidad está pensada para el uso del dispositivo por parte del encargado de soporte en varias empresas cliente, ya que puede necesitar conectarse a cualquiera de ellas en cualquier momento usando su dispositivo móvil.

Por tanto, el uso de un gestor de perfiles en el dispositivo y por medio de este software, le permite mantener actualizados y en validez en todo momento cualquier certificado de los perfiles de PKI que tenga configurados.

Otras funcionalidades internas futuras en cuanto a la seguridad de los datos almacenados en el dispositivo Android podría ser la adaptación de la librería interna de la empresa de securización de configuración a Java 1.6 (y por lo tanto, hacerlo compatible en Android), o especificar los distintos niveles de Log hacia afuera (para mejorar el soporte sobre la aplicación).

Bibliografía

[AL02] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations (2nd Edition)*. Addison-Wesley Professional, 2002.

[Ali09] Jorge Blasco Alis. *Proyecto Final de Carrera - Antecedentes y perspectivas de estudio en historia de la Criptografía*. Universidad Carlos III de Madrid, 2009.

[Bid04] Hossein Bidgoli. *The Internet Encyclopedia. Volume 3*. Wiley & Sons, 2004.

[Cop] Jack Copeland. BBC Article: *Alan Turing: The codebreaker who saved 'millions of lives'*. Última visita: 27-02-2014.

[dL11] César Guzmán de Lapuente. *Proyecto Final de Carrera - Implantación de un sistema de certificados*. Universidad Politécnica de Cataluña, 2011.

[EA03] Jon Edney and William A. Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley Professional, 2003.

[iC11] Lluís Moran i Capdevila. *Proyecto Final de Carrera - Autoritat de certificació PKI amb serveis en línia*. Universitat Oberta de Catalunya, 2011.

[KTD11] Andre Karamanian, Srinivas Tenneti, and Francois Dessart. *PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks (Networking Technology: Security)*. Cisco Press, 2011.

[OPR+12] Gabriel Díaz Orueta, Francisco Mur Pérez, Elio Sancristóbal Ruiz, Manuel Alonso Castro Gil, and Juan Peire Arroba. *Seguridad en las comunicaciones y en la información*. Universidad Nacional de Educación a Distancia (UNED), 2012.

[Oso10] Juan Manuel Alor Osorio. *Proyecto Final de Carrera - Evaluación de la herramienta EJBCA para un Prestador de Servicios de Certificación.* Universidad Politécnica de Cataluña, 2010.

[Sta10] William Stallings. *Cryptography and Network Security: Principles and Practice (5th Edition).* Prentice Hall, 2010.

[W1] Guía de seguridad de las TIC. *CCN-STIC-453 (CNI - Ministerio de Defensa).* `http://goo.gl/zi2Gpf`. Última visita: 27-02-2014.

[W10] Blog de Nikolay Elenkov. *ICS Trust Store Implementation.* . `{http://goo.gl/vYEyBi}`. Última visita: 27-02-2014.

[W11] Blog de Nikolay Elenkov. *Using a Custom Certificate Trust Store on Android.* . `{http://goo.gl/Q9NI85}`. Última visita: 27-02-2014.

[W12] Blog de Nikolay Elenkov. *Storing Application secrets in Android's credential storage.* . `{http://goo.gl/ris3ot}`. Última visita: 27-02-2014.

[W13] Blog de Nikolay Elenkov. *Certificate blacklisting in Jelly Bean.* . `{http://goo.gl/kGXRLO}`. Última visita: 27-02-2014.

[W14] *BouncyCastle* - Especificaciones. . `{http://www.bouncycastle.org/specifications.html}`. Última visita: 27-02-2014.

[W15] Blog de Maxim Porges. *Configurinng Tomcat SSL Client/Server Authentication* . `{http://goo.gl/uHxGOl}`. Última visita: 27-02-2014.

[W16] Blog *Chuidiang*: Ejemplo sencillo de servlet. . `{http://goo.gl/TUl7oO}`. Última visita: 27-02-2014.

[W17] Blog *NetworkLessons.com: EAP-TLS with Server 2008 SCEP for Apple Devices* . `{http://goo.gl/izCsH7}`. Última visita: 27-02-2014.

[W18] Blog *TechNet España*: Despliegue de Certificados para IPADs vía NDES y Autenticación a una Red Wifi 802.1x . `{http://goo.gl/pKxD5q}`. Última visita: 27-02-2014.

[W19] *iOS Developer Library: Over-the-Air Profile Delivery Concepts* . `{http://goo.gl/RWYb8q}`. Última visita: 27-02-2014.

[W2] Blog Linux y Libertad - *¿Fue una buena idea usar AES256 con el archivo INSURANCE de Wikileaks?* `http://goo.gl/DaQocr`. Última visita: 27-02-2014.

[W20] *Documentación BB:* Flujo de datos del protocolo de inscripción de certificados simple (SCEP) de BlackBerry 10 OS . {http://goo.gl/jWd0yA}. Última visita: 27-02-2014.

[W21] *CSS Networks: The Use of the Simple Certificate Enrollment Protocol (SCEP) and Untrusted Devices* . {http://goo.gl/4ouH3f}. Última visita: 27-02-2014.

[W22] Craig Gentry. *Certificate-Based Encryption and the Certificate Revocation Problem* . {http://goo.gl/ieHLmP}. Última visita: 27-02-2014.

[W23] *Cisco Networks: Simple Certificate Enrollment Protocol Overview* . {http://goo.gl/AIhPhg}. Última visita: 27-02-2014.

[W24] Microsoft TechNote: *Network Device Enrollment Service (NDES) Whitepaper* . {http://goo.gl/aNxJk8}. Última visita: 27-02-2014.

[W25] Simple Certificate Enrollment Protocol:*draft-nourse-scep-23* . {https://tools.ietf.org/html/draft-nourse-scep-23}. Última visita: 27-02-2014.

[W26] Android Jelly Bean's Face Unlock "Liveness CheckÇircumvented With Simple Photo Editing . {http://goo.gl/NMFY58}. Última visita: 27-02-2014.

[W27] StackOverflow - How to install Trusted Certificate on Android Store . {http://goo.gl/Kjhlvz}. Última visita: 27-02-2014.

[W3] Dag Arne Osvik, Adi Shamir and Eran Tromer. *Cache Attacks and Countermeasures: the Case of AES (Extended Version*). Última visita: 27-02-2014.

[W4] Atsuko Miyaji, Masao Nonaka and Yoshinori Takii. *Improved Correlation Attack on RC5.* http://goo.gl/zKaOaN. Última visita: 27-02-2014.

[W5] 3GPP TSG SA WG3 Security. *CMP and CMC Comparison.* http://goo.gl/GN087S. Última visita: 27-02-2014.

[W6] Technote: *Security Providers in Java* . {http://goo.gl/xxO69a}. Última visita: 27-02-2014.

[W7] Cisco Networks TechNote: *Configure HTTPS Support for ISE SCEP Integration.* . {http://goo.gl/gQmxid}. Última visita: 27-02-2014.

[W8] Blog de Nikolay Elenkov. *Using the ICS KeyChain API.* . `{http://goo.gl/QKX3xA}`. Última visita: 27-02-2014.

[W9] Blog de Nikolay Elenkov. *ICS Credential Storage Implementation.* . `{http://goo.gl/tHiJOl}`. Última visita: 27-02-2014.

Anexo A

Glosario

- **3DES**: Triple DES.
  - Algoritmo de cifra que realiza 3 veces DES.
- **ADB**: Android Debug Bridge.
  - Línea de comandos que permiten la conexión con un emulador o dispositivo Android.
- **AD:CS**: Active Directory Certificate Services.
  - Tecnología de control de acceso e identidad que proporciona servicios de creación y administración de certificados y claves en sistemas operativos Windows y Windows Server.
- **AES**: Advanced Encryption Standard.
  - Algoritmo de cifrado simétrico adoptado como estándar por el gobierno de EE.UU.
- **API**: Application Programming Interface.
  - Conjunto de funciones que proporciona una librería para su uso como capa de abstracción.
- **BC**: BouncyCastle.
  - Librería criptográfica que permite realizar operaciones criptográficas para Java y C++.

- **BYOD**: Bring Your Own Device.
  - Tendencia por la que se permite el uso de dispositivos personales como si fueran corporativos.
- **CA**: Certificate Authority.
  - Entidad en la que confían todos los miembros de una PKI.
- **CDP**: CRL Distribution Point.
  - Punto de enlace a la CRL actual proporcionada por la CA.
- **CGI**: Common Gateway Interface.
  - Tecnología que permite a un cliente obtener datos de un programa ejecutado en un servidor Web.
- **CMC**: Certificate Management over CMS.
  - Protocolo estándar de gestión de certificados.
- **CMP**: Certificate Management Protocol.
  - Protocolo estándar de obtención de certificados X509.
- **CRL**: Certificate Revocation List.
  - Lista de números de serie de certificados revocados mantenida por la CA.
- **CSR**: Certificate Signing Request.
  - Petición de firma de certificado (solicitud de certificación que se manda a la CA).
- **DES**: Data Encryption Standard.
  - Algoritmo de cifrado simétrico por bloques.
- **DPC**: Declaración de Políticas de Certificación.
  - Documento que define los procedimientos y mecanismos que han de seguirse a la hora de obtener certificados digitales en el entorno de una PKI.
- **Draft**: Borrador de Estándar

  - Documento de definición de una propuesta de estándar que se envía a una asociación internacional encargada de la definición de estándares.

- **DSA**: Digital Signature Algorithm.
  - Algoritmo estándar de generación de firma digital.
- **ECC**: Elliptical Curve Cryptography.
  - Criptografía de clave pública que usa la estructura algebraica de curvas elípticas en espacio finito.
- **EST**: Enrollment Secure Transport.
  - Protocolo estándar de obtención y gestión de certificados digitales.
- **FNMT**: Fábrica Nacional de Moneda y Timbre.
  - Entidad adscrita al Ministerio de Hacienda y dedicada a la fabricación de monedas, billetes y sellos. También proporciona servicios de certificación.
- **HTTP**: Hyper-Text Transport Protocol.
  - Protocolo en las transacciones que tienen lugar en la World Wide Web.
- **HTTPS**: Secure Hyper-Text Transport Protocol.
  - Implementación del protocolo HTTP sobre SSL
- **ICT**: Information and Communication Technology.
  - Ver TIC.
- **IDEA**: International Data Encryption Algorithm.
  - Algoritmo de clave simétrica que realiza cifrado por bloques.
- **IETF**: Internet Engineering Task Force.
  - Asociación internacional que propone y realiza la especificación de estándares orientados a Internet.
- **IIS**: Internet Information Server.
  - Servidor Web desarrollado por Microsoft e incluido en sistemas operativos Windows Server.

- **IOS**: iPhone Operative System.
  - Sistema operativo utilizado en dispositivos iPad e iPhone de Apple.
- **IPSEC**: Internet Protocol SECurity.
  - Conjunto de protocolos encargados del cifrado y/o autenticación de los paquetes involucrados en una comunicación de datos.
- **JAR**: Java ARchive.
  - Contenedor de clases y librerías desarrolladas en lenguaje Java que permite la ejecución de aplicaciones.
- **JCA**: Java Cryptography Architecture.
  - Framework de programación que proporciona primitivas de criptografía en el lenguaje Java.
- **JCE**: Java Cryptography Extension.
  - Extensión a JCA para la realización de cifrado, generación de claves o firma.
- **JDK**: Java Development Kit.
  - SDK en lenguaje Java.
- **JKS**: Java KeyStore.
  - Implementación de almacén de credenciales que se usa por defecto en una JRE.
- **JRE**: Java Runtime Environment.
  - Máquina Virtual en la que se ejecutan los programas realizados en lenguaje Java.
- **JSP**: Java Server Pages.
  - Tecnología Java que permite la generación de páginas Web mediante código Java ejecutado en servidor.
- **MD5**: Message-Digest Algorithm 5.
  - Algoritmo estándar de cálculo de hashes (verificación de integridad).

- **MMC**: Memory Management Console.
  - Herramienta Windows de administración de servidor.
- **NDES**: Network Device Enrollment Service.
  - Implementación del protocolo SCEP realizada por Microsoft e incluida en Windows Server 2008.
- **NIST**: National Institute of Standards and Technology
  - Agencia federal estadounidense encargada de la definición y promoción de estándares tecnológicos e industriales.
- **OCSP**: Online Certificate Status Protocol.
  - Protocolo de comprobación del estado de revocación de certificados.
- **OTP**: One-Time Password.
  - Método de autenticación basado en el uso de una contraseña de un único uso.
- **PBKDF2**: Password-Based Key Derivation Format 2.
  - Función de derivación de claves que forma parte del estándar PKCS#5. Aplica iterativamente una función pseudoaleatoria a una contraseña con un valor sal (bits aleatorios) para derivar otra clave que puede ser usada como simétrica.
- **PEM**: Privacy Enhanced Mail.
  - Formato de fichero que contiene un certificado o clave codificado en Base64.
- **PGP**: Pretty Good Privacy
  - Programa diseñado para proteger la información comunicada a través de Internet por medio del uso de criptografía de clave pública.
- **PIN**: Personal Identification Number
  - Número de identificación personal.
- **PKCS**: Public Key Cryptography Standard.

  - Conjunto de estándares usados en criptografía de clave pública y definidos por RSA Security.
    - **PKCS#7:** Estándar de mensaje criptográfico. Usado para cifrar o autenticar un mensaje (RFC 2315).
      `https://www.ietf.org/rfc/rfc2315.txt`
    - **PKCS#8:** Estándar de información y sintaxis de la clave privada (RFC 5208).
      `https://www.ietf.org/rfc/rfc5208.txt`
    - **PKCS#10:** Estándar de solicitud de certificación. (RFC 2986).
      `https://www.ietf.org/rfc/rfc2986.txt`

- **PKI**: Public Key Infrastructure.
  - Conjunto de definiciones, dispositivos y políticas que componen un conjunto de confianza para el uso de certificados digitales.
- **PKIX**: X509 Public Key Infrastructure.
  - PKI que utiliza certificados basados en el estándar X.509.
- **RA**: Registry Authority.
  - Entidad de una PKI que revisa las solicitudes de certificación enviadas a la CA.
- **RC5**: Rivest Cipher 5
  - Algoritmo de cifra basada en bloques con uso de clave simétrica.
- **RFC**: Request for Comments
  - Propuesta de estándar definida por el IETF. Las analizadas en el documento son las siguientes:
    - RFC 2315 : `https://www.ietf.org/rfc/rfc2315.txt`
    - RFC 2510 : `https://www.ietf.org/rfc/rfc2510.txt`
    - RFC 2511 : `https://www.ietf.org/rfc/rfc2511.txt`
    - RFC 2560 : `https://www.ietf.org/rfc/rfc2560.txt`
    - RFC 2797 : `https://www.ietf.org/rfc/rfc2797.txt`
    - RFC 2986 : `https://www.ietf.org/rfc/rfc2986.txt`
    - RFC 3161 : `https://www.ietf.org/rfc/rfc3161.txt`
    - RFC 3280 : `https://www.ietf.org/rfc/rfc3280.txt`

    - RFC 3647 : https://www.ietf.org/rfc/rfc3647.txt
    - RFC 5208 : https://www.ietf.org/rfc/rfc5208.txt
    - RFC 5272 : https://www.ietf.org/rfc/rfc5272.txt
    - RFC 5280 : https://www.ietf.org/rfc/rfc5280.txt
    - RFC 5867 : https://www.ietf.org/rfc/rfc5867.txt
    - RFC 6818 : https://www.ietf.org/rfc/rfc6818.txt
    - RFC 7030 : https://www.ietf.org/rfc/rfc7030.txt

- **RO**: Registry Operator.
    - Persona física que se encarga de verificar peticiones de certificación.
- **RSA**: RivestShamirAdelman.
    - Algoritmo de clave pública que permite la generación de pares de claves pública-privada.
- **SoC**: System on Chip.
    - Dispositivo que integra varios componentes electrónicos en un único chip. Puede incluir circuitos de señal digital, circuitos de señal analógica y módulos de comunicación por radio-frecuencia.
- **SCEP**: Simple Certificate Enrollment Protocol.
    - Protocolo no estándar de gestión y obtención automática de certificados digitales.
- **SDK**: Source Development Kit.
    - Colección de herramientas de programación para la realización de software en un lenguaje determinado.
- **SHA**: Secure Hash Algorithm.
    - Familia estándar criptográfica de realización de funciones hash.
- **SSL**: Secure Sockets Layer.
    - Protocolo que usa funciones criptográficas para garantizar una comunicación segura en Interner.
- **TIC**: Tecnologías de la Información y la Comunicación

  - En lo relativo a las empresas, aquellas pertenecientes al sector tecnológico de las telecomunicaciones y al del desarrollo de productos informáticos.

- **TLS**: Transport Layer Security.
  - Ver SSL.
- **TSA**: Time Stamp Authority.
  - Entidad de una PKI que aplica sellos de tiempo fiables.
- **TSP**: Time Stamp Protocol.
  - Protocolo criptográfico de certificación de sellos de tiempo mediante certificados X509 en una PKI.
- **UIT-T**: ITU Telecommunication Standardization Sector.
  - Sección de estandarización para telecomunicaciones de la ITU (*International Telecommunication Union*).
- **URI**: Uniform Resource Identifier.
  - Conjunto de caracteres que identifican el nombre de un recurso Web.
- **VA**: Validation Authority.
  - Entidad encargada de la comprobación de la validez de certificados digitales.
- **VPN**: Virtual Private Network.
  - Definición de la extensión de uso de una red privada a través de Internet.
- **WEP**: Wired Equivalent Privacy
  - Sistema de cifrado de comunicaciones inalámbricas incluido en el estándar IEEE 802.11
- **WiFi**: Wireless Fidelity
  - Conjunto de normas y métodos que definen una forma de conexión inalámbrica entre dispositivos electrónicos.
- **X509**: Formato estándar digital de certificado.
  - Estándar ITU-T para la definición de certificados digitales para su uso en una PKI.

Anexo B

Manual de usuario de la aplicación Windows

B.1. Ejecución inicial

Una vez instalado, el programa puede ser iniciado por medio del icono de escritorio, o bien en el siguiente reinicio del ordenador (que realizará una comprobación rutinaria de los certificados instalados).

Su ejecución provoca que se instale en la barra de tareas de Windows un icono como el siguiente (Figura B.1):

Figura B.1: Icono Windows en barra de tareas

Al iniciar la ejecución del programa, se realiza una primera comprobación de los certificados instalados en el dispositivo. Se muestran los mensajes de conexión al servidor y los del estado de la comprobación.

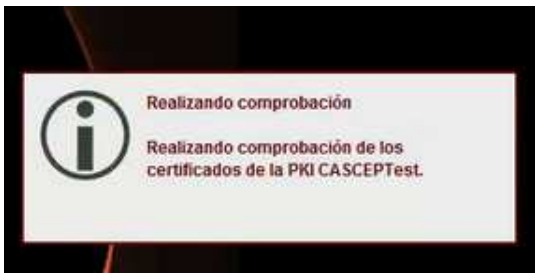

Figura B.2: Ticker Windows: Comienzo comprobación períodica

En su primera ejecución, el programa realiza una primera comprobación de los certificados instalados (Figura B.2). Al no estar instalados, mostrará el mensaje de notificación de instalación.

Cualquier error en la conexión con el servidor al realizar las operaciones del protocolo se verán reflejadas en las notificaciones de siguiente reintento (Figura B.4). Cuando se ha alcanzado el reintento final, se muestra la notificación correspondiente (Figura B.3) de cuenta atrás antes de volver a intentarlo automáticamente. Si se pulsa en el mensaje, se volverá a reintentar la operación que provocó el fallo.

Figura B.3: Ticker Windows: Error definitivo de conexión

Figura B.4: Ticker Windows: Cuenta atrás de reintento de conexión.

B.2. Renovación previa de certificado de dispositivo

El programa realiza una programación de las tareas de renovación del certificado del dispositivo (Figura B.5). Esta tarea realiza la solicitud al servidor SCEP tal y como haya sido configurada previamente.

Los posibles resultados de la solicitud son los siguientes:

- Solicitud correcta y certificado reinstalado (Figura B.6).
- Error en la solicitud enviada (Figura B.7).
- Error al reinstalar el certificado de dispositivo (Figura B.8).
- Solicitud de renovación no soportada (Figura B.9).

Figura B.5: Ticker Windows: Inicio de renovación de certificado del dispositivo.

Figura B.6: Ticker Windows: Solicitud de renovación del certificado de dispositivo correcta.

Figura B.7: Ticker Windows: Error en la solicitud de renovación.

Figura B.8: Ticker Windows: Error en la reinstalación del certificado de dispositivo.

Figura B.9: Ticker Windows: Solicitud de renovación no soportada.

B.3. Roll-Over previo de certificados

El cliente SCEP realiza una programación de la realización del roll-over previo a la caducidad del certificado de la CA (Figura B.10). Esta tarea realiza la solicitud

al servidor SCEP tal y como haya sido configurada previamente.

Los posibles resultados de la solicitud son los siguientes:

- Solicitud correcta y certificados reinstalados (Figura B.13).
  - En este caso, el sistema operativo pide la desinstalación del certificado anteriormente instalado (Figura B.11), y la instalación del nuevo (Figura B.12).
- Solicitud de roll-over de certificados no soportada (Figura B.14).
- Error al reinstalar los certificados renovados (Figura B.15).

Figura B.10: Ticker Windows: Inicio de roll-over de certificados.

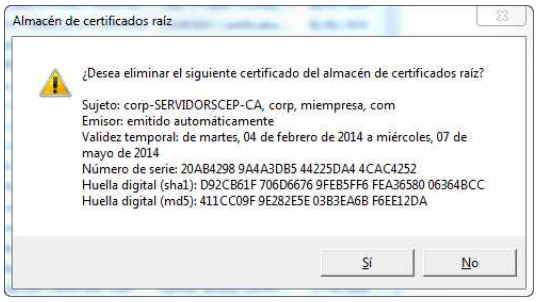

Figura B.11: Ticker Windows: Desinstalar certificado de CA.

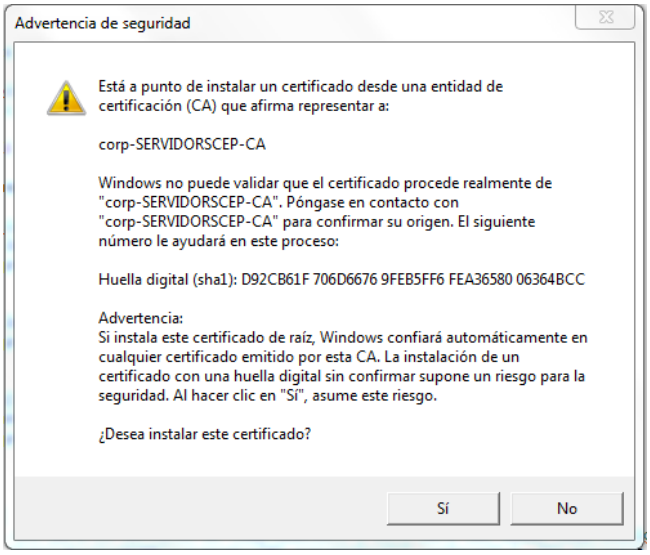

Figura B.12: Ticker Windows: Instalación certificado CA.

Figura B.13: Ticker Windows: Nuevos certificados instalados tras roll-over correcto.

Figura B.14: Ticker Windows: Roll-over no soportado por el servidor SCEP.

Figura B.15: Ticker Windows: Error en la reinstalación de los certificados renovados.

B.4. Reinstalación de certificado de dispositivo tras caducidad o revocación

En caso de que la comprobación del certificado del dispositivo haya ocurrido después de la fecha de caducidad del certificado de dispositivo, o de que el certificado haya sido revocado sin que el cliente SCEP lo haya notificado; el cliente SCEP realiza la solicitud de recertificación del mismo tal y como si fuera una nueva instalación (Figura B.16).

Figura B.16: Ticker Windows: Inicio de reinstalación de certificado tras caducidad/revocación.

Los posibles resultados de la solicitud son los siguientes:

- Solicitud correcta y certificado reinstalado (Figura B.17).
- Solicitud de recertificación del dispositivo errónea (Figura B.18).

- Error al reinstalar los certificados renovados (Figura B.19).

Figura B.17: Ticker Windows: Recertificación de dispositivo correcta tras caducidad.

Figura B.18: Ticker Windows: Solicitud de recertificación errónea.

Figura B.19: Ticker Windows: Error al instalar el nuevo certificado de dispositivo.

B.5. Reinstalación de certificados tras caducidad o revocación del certificado de la CA

En caso de que la comprobación del certificado de la CA haya ocurrido después de la fecha de caducidad del certificado de dispositivo (Figura B.20), o de que el certificado haya sido revocado sin que el cliente SCEP lo haya notificado (Figura B.21); el cliente SCEP realiza la solicitud del nuevo certificado de CA y la recertificación del dispositivo tal y como si fuera una nueva instalación.

Figura B.20: Ticker Windows: Inicio de reinstalación de certificados tras caducidad de certificado de CA.

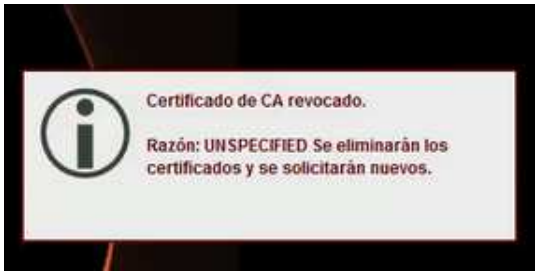

Figura B.21: Ticker Windows: Inicio de reinstalación de certificados tras revocación de certificado de CA.

Los posibles resultados de la solicitud son los siguientes:

- Solicitud correcta y certificados reinstalados (Figura B.22).
- Error en la solicitud de recertificación (Figura B.23).
- Error al reinstalar los certificados renovados (Figura B.24).

Figura B.22: Ticker Windows: Recertificación de dispositivo y CA correcta tras caducidad/revocación.

Figura B.23: Ticker Windows: Solicitud de recertificación tras caducidad de CA errónea.

Figura B.24: Ticker Windows: Error al instalar el nuevo certificado de dispositivo y de CA.

B.6. Comprobación del estado de los certificados

Esta es la tarea principal del cliente SCEP de escritorio. Al iniciar el ordenador, al iniciar el cliente, al pedirse por demanda (Figura B.1), o al vencer el temporizador interno de comprobación (configurable previamente); se realiza la comprobación de los certificados instalados en el dispositivo (Figura B.2).

Se distinguen 3 casos principales, comprobados en dicho orden:

- Certificado de CA no instalado
- Certificado de dispositivo no instalado
- Comprobar certificados instalados de CA y dispositivo

B.6.1. Certificado de CA no instalado

Si el certificado de la CA no está instalado (Figura B.25), se pide su envío al servidor SCEP. Una vez enviado, se solicitará al usuario la aceptación de la instalación de dicho certificado en el almacén de "Entidades raíz de confianza" (Figura B.12). El usuario debe pulsar el botón "Sí", para finalizar la instalación.

Figura B.25: Ticker Windows: Certificado de CA no instalado.

Los posibles errores que se pueden dar durante el procedimiento de obtención e instalación del certificado de CA en el dispositivo son:

- Error en la instalación del certificado (Figura B.26).
- Error en la comprobación del hash preprovisionado del certificado (Figura B.27).

- Sólo se da si previamente se ha configurado la comprobación del hash del certificado de la CA.

Figura B.26: Ticker Windows: Error en la instalación del certificado de la CA.

Figura B.27: Ticker Windows: Error en la verificación del hash preprovisionado del certificado de CA.

Tras la instalación del certificado de la CA, no se comprueba su estado, ya que se supone que el certificado que envía el servidor SCEP es el certificado actual y válido de la CA. Se programan las tareas de roll-over previo y caducidad.

B.6.2. Certificado de dispositivo no instalado

Si el certificado del dispositivo no está instalado, la casuística se divide en 2 casos:

- El dispositivo solicitó un certificado emitido anteriormente (Figura B.31).

- En este caso, y por alguna razón, el certificado del dispositivo ha sido desinstalado del almacén de certificados del sistema operativo. El cliente SCEP guarda el número de serie del último certificado emitido y pide su reemisión al servidor SCEP.

  Si no hay ningún error durante la transacción, se muestra el mensaje de éxito correspondiente (Figura B.13).

  Si hay algún error, el cliente solicita al servidor SCEP la emisión de un nuevo certificado, tal y como si fuera la primera vez.

- El dispositivo no ha solicitado la emisión de un certificado con anterioridad.

  - En este caso, se considera que el cliente SCEP ha sido ejecutado por primera vez y no tiene ningún dato asociado a un certificado emitido anteriormente. Por ello, solicita la emisión de un nuevo certificado al servidor SCEP.
    Si ha ocurrido algún error, se muestra el mensaje correspondiente. Si no ha ocurrido ningún error durante la transacción, se muestran los posibles resultados de la operación
    - Operación aprobada: el certificado ha sido emitido e instalado en el dispositivo (Figura B.28).
    - Operación pendiente de aprobación manual: falta una aprobación manual de emisión del certificado, y el cliente SCEP instalará el certificado en la siguiente comprobación (Figura B.29).
    - Operación rechazada: la CA no ha aprobado la solicitud de certificación (Figura B.30).

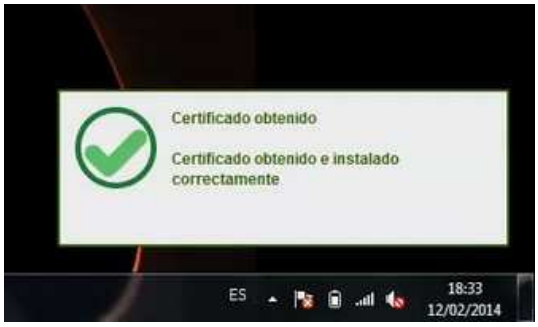

Figura B.28: Ticker Windows: Solicitud de certificación correcta.

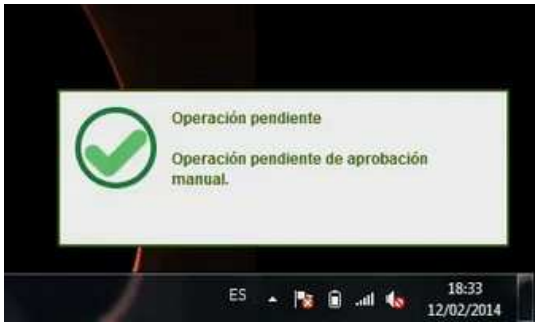

Figura B.29: Ticker Windows: Solicitud de certificación marcado como pendiente.

Figura B.30: Ticker Windows: Solicitud de certificación rechazada.

Figura B.31: Ticker Windows: Certificado de dispositivo ya emitido.

B.6.3. Comprobar certificados instalados de CA y dispositivo

Si tanto el certificado de la CA (Figura B.32) como el del dispositivo (Figura B.33) están instalados con anterioridad a la comprobación (, se realiza una comprobación de su estado (Ver B.5). En esta comprobación se pueden dar 5 resultados correctos:

- Certificado de CA caducado (Figura B.20).
  - Como consecuencia, se arranca el tratamiento de la solicitud de roll-over, o la reinstalación de certificados nuevos (tanto de CA como del dispositivo).
- Certificado de CA revocado (Figura B.21).
  - Como consecuencia, se arranca la reinstalación de certificados nuevos (tanto de CA como del dispositivo).
- Certificado de dispositivo caducado (Figura B.16).
  - Como consecuencia, se instala un certificado anteriormente renovado pero no instalado, o se vuelve a solicitar la emisión de un certificado como si fuera la primera vez.
- Certificado de dispositivo revocado (Figura B.35).
  - Mostrando la razón de revocación. Como consecuencia, se vuelve a solicitar la emisión de un certificado como si fuera la primera vez.
- Certificado de CA y de dispositivo válidos (Figura B.34).
  - Mostrando la razón de revocación. Como consecuencia, se vuelve a solicitar la emisión de un certificado como si fuera la primera vez.

Figura B.32: Ticker Windows: Comprobación de certificado de CA instalado.

Figura B.33: Ticker Windows: Comprobación de certificado de dispositivo instalado.

Figura B.34: Ticker Windows: Certificado de dispositivo y de CA válidos tras comprobación.

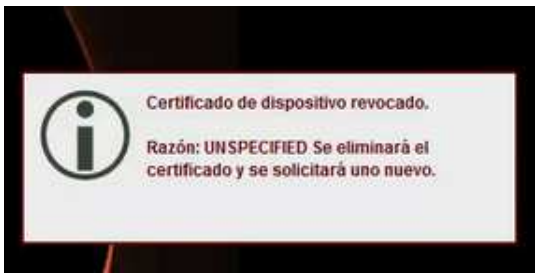

Figura B.35: Ticker Windows: Certificado de dispositivo revocado sin razón específica.

Si no se puede realizar alguna de las operaciones se obtienen los siguientes errores:

- CRL no accesible: no se puede comprobar el estado de revocación del certificado de CA o del dispositivo (Figura B.36).
- Error en la instalación del certificado de dispositivo (Figura B.24) o en el de la CA (Figura B.26).
- Error en la solicitud (Figura B.23).

Figura B.36: Ticker Windows: Error en la consulta de la CRL.

Anexo C

Manual de usuario de la aplicación Android

C.1. Ejecución inicial

Una vez instalado, el programa puede ser iniciado por medio del icono que se verá reflejado en el *Launcher* (Figuras C.1 y C.2), o bien en el siguiente reinicio del terminal (que realizará una comprobación rutinaria de los certificados instalados).

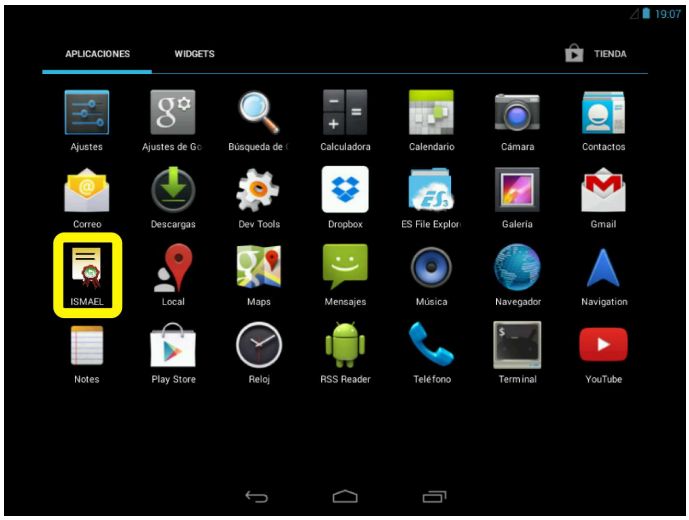

Figura C.1: Ejecución software Android por Launcher

Figura C.2: Activity inicial

En su primera ejecución, el programa realiza una primera comprobación de los certificados instalados. Al no estar instalados, mostrará el mensaje de notificación de instalación (Figuras C.17 y C.18).

Cualquier error en la conexión con el servidor al realizar las operaciones del protocolo se verán reflejadas en las notificaciones de siguiente reintento (Figuras C.5 y C.6). Cuando se ha alcanzado el reintento final, se muestra la notificación correspondiente (Figuras C.3 y C.4) de cuenta atrás antes de volver a intentarlo automáticamente. Si se pulsa en el mensaje, se volverá a reintentar la operación que provocó el fallo.

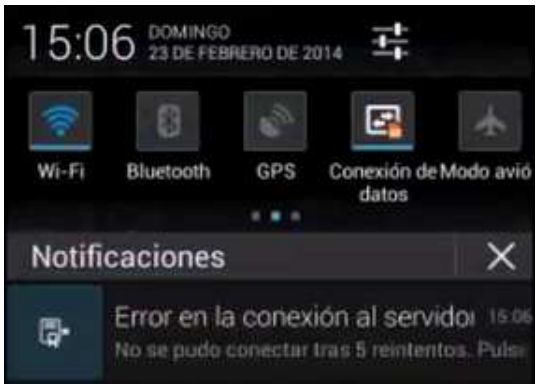

Figura C.3: Notification Android: Error definitivo de conexión

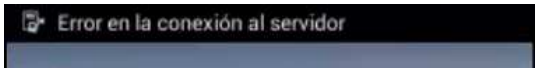

Figura C.4: Ticker Android: Error definitivo de conexión

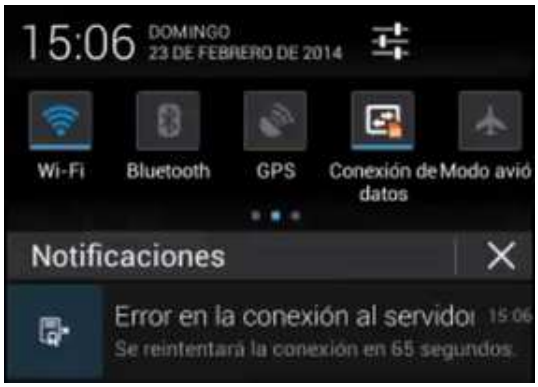

Figura C.5: Notification Android: Cuenta atrás de reintento de conexión.

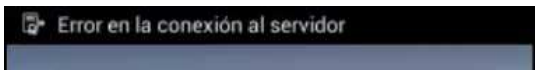

Figura C.6: Ticker Android: Cuenta atrás de reintento de conexión.

C.2. Instalación de certificados

Si se hace click en la notificación de instalación de certificados, el cliente SCEP se pone en contacto con el servidor y le solicita en primer lugar el certificado de la CA para poder realizar la inscripción (*enrollment*) del dispositivo. Durante dicho procedimiento se pueden producir las siguientes notificaciones derivadas:

- Error en la obtención del certificado de CA (Figuras C.7 y C.8).
- Error en la solicitud de certificación del dispositivo (Figuras C.11 y C.12).
- Error en la instalación del certificado (Figuras C.13 y C.14).
- Error en la generación de claves (Figuras C.15 y C.16).
- Instalación correcta de certificados (Figuras C.9 y C.10).

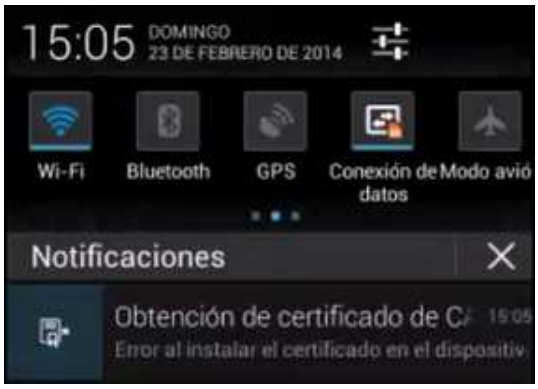

Figura C.7: Notification Android: Error en la obtención del certificado de CA

Figura C.8: Ticker Android: Error en la obtención del certificado de CA

Figura C.9: Notification Android: Instalación correcta de certificados

Figura C.10: Ticker Android: Instalación correcta de certificados

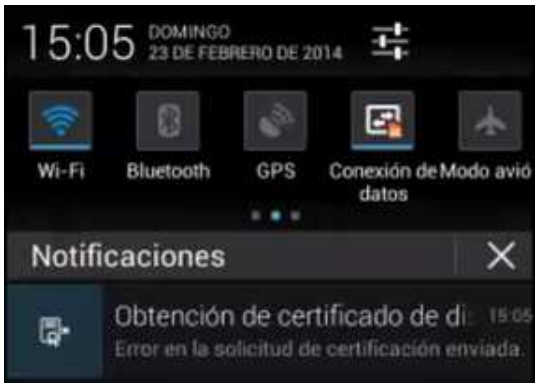

Figura C.11: Notification Android: Error en la solicitud de certificación del dispositivo

Figura C.12: Ticker Android: Error en la solicitud de certificación del dispositivo

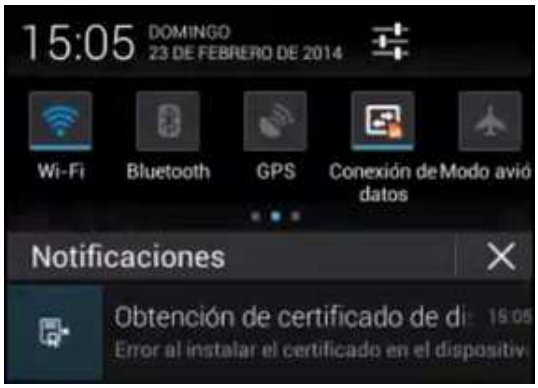

Figura C.13: Notification Android: Error en la instalación del certificado

Figura C.14: Ticker Android: Error en la instalación del certificado

Figura C.15: Notification Android: Error en la generación de claves

Figura C.16: Ticker Android: Error en la generación de claves

En el caso de la instalación correcta de certificados, se muestra como parte de la instalación de certificados una pantalla de introducción de contraseña del certificado (Figura C.19). Por defecto es vacío, con lo que se pulsa en el botón `Aceptar`. En

la siguiente pantalla se observa el resumen de la instalación (Figura C.20): nombre del perfil (Ver Fichero de Configuración), y su contenido: certificado de usuario, certificado de CA y clave de usuario. Pulsar en `Aceptar` para finalizar la instalación de los certificados en el dispositivo.

Figura C.17: Notification Android: Certificados no instalados

Figura C.18: Ticker Android: Certificados no instalados

Figura C.19: Android: Introducción de contraseña de instalación

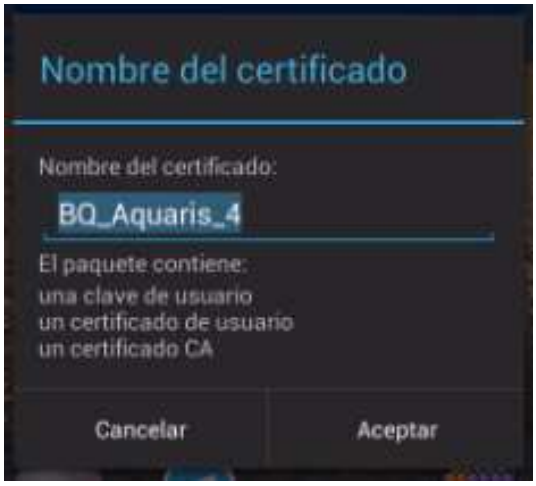

Figura C.20: Android: Instalación de certificados en dispositivo

C.3. Comprobación de certificados

Si tanto el certificado de la CA como el del dispositivo estuviesen instalados, se realiza una comprobación del estado de revocación y la fecha de caducidad de los mismos. Durante este procedimiento, la aplicación mostrará alguno de los siguientes mensajes y notificaciones, según la situación:

- Error al realizar la comprobación del estado del certificado de CA (Figuras C.21 y C.22).
- Certificado de CA revocado (Figuras C.23 y C.24).
- Certificado de CA caducado (Figuras C.25 y C.26).
- Certificado de CA próximo a caducar (Figuras C.27 y C.28).
- Error al realizar la comprobación del estado del certificado de dispositivo (Figuras C.29 y C.30).
- Certificado de dispositivo revocado (Figuras C.31 y C.32).
- Certificado de dispositivo caducado (Figuras C.33 y C.34).
- Certificado de dispositivo próximo a caducar (Figuras C.35 y C.36).
- Certificados verificados y válidos (Figuras C.37 y C.38).

Figura C.21: Notification Android: Error en comprobación del estado del certificado de CA

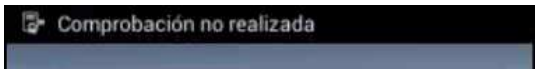

Figura C.22: Ticker Android: Error en comprobación del estado del certificado de CA

Figura C.23: Notification Android: Certificado de CA revocado

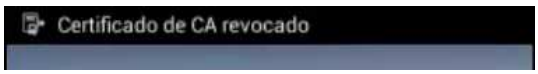

Figura C.24: Ticker Android: Certificado de CA revocado

Figura C.25: Notification Android: Certificado de CA caducado

Figura C.26: Ticker Android: Certificado de CA caducado

Figura C.27: Notification Android: Certificado de CA próximo a caducar

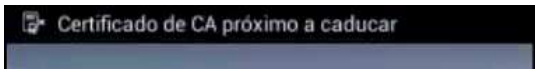

Figura C.28: Ticker Android: Certificado de CA próximo a caducar

Figura C.29: Notification Android: CRL no accesible con comprobación obligatoria

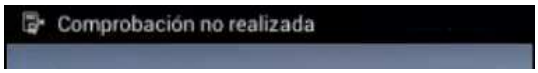

Figura C.30: Ticker Android: CRL no accesible con comprobación obligatoria

Figura C.31: Notification Android: Certificado de dispositivo revocado

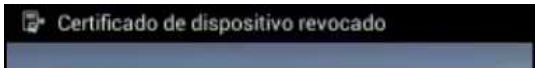

Figura C.32: Ticker Android: Certificado de dispositivo revocado

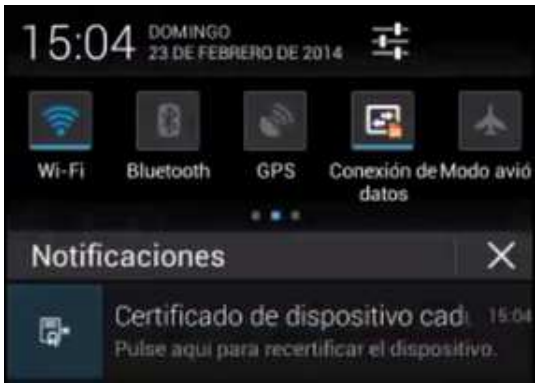

Figura C.33: Notification Android: Certificado de dispositivo caducado

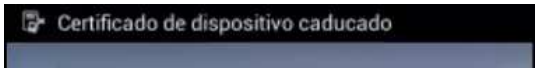

Figura C.34: Ticker Android: Certificado de dispositivo caducado

Figura C.35: Notification Android: Certificado de dispositivo próximo a caducar

Figura C.36: Ticker Android: Certificado de dispositivo próximo a caducar

Figura C.37: Notification Android: Certificados verificados y válidos

Figura C.38: Ticker Android: Certificados verificados y válidos

C.4. Renovación previa de certificados

Si se hace click en alguna de las notificaciones de renovación previa de certificados (tanto de CA -Figuras C.27 y C.28, como de dispositivo -Figuras C.35 y C.36), se realiza la solicitud correspondiente de renovación de certificados inmediata. Su resultado puede ser uno de los siguientes:

- Renovación previa correcta (Figuras C.39 y C.40).
- Error en solicitud de renovación previa (Figuras C.41 y C.42).
- Error interno en renovación previa (Figuras C.43 y C.44).

Figura C.39: Notification Android: Renovación previa correcta

Figura C.40: Ticker Android: Renovación previa correcta

Figura C.41: Notification Android: Error en solicitud de renovación previa

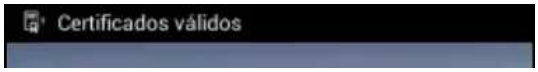

Figura C.42: Ticker Android: Error en solicitud de renovación previa

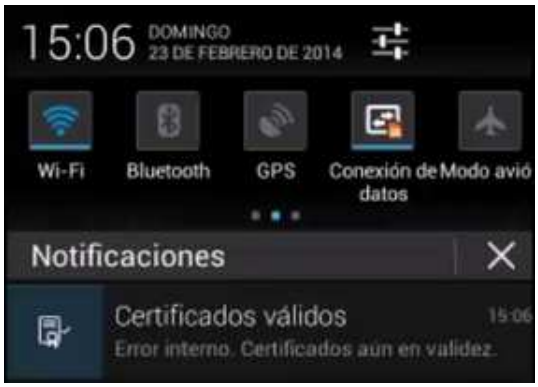

Figura C.43: Notification Android: Error interno en renovación previa

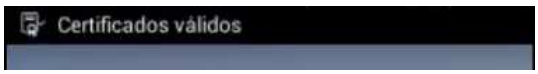

Figura C.44: Ticker Android: Error interno en renovación previa

C.5. Reinstalación de certificados

Al pulsar sobre la notificación correspondiente, se realiza una acción asociada. Si se pulsa en la notificación de reinstalación de certificados tras caducidad (Figuras C.25 y C.26) o revocación (Figuras C.23 y C.24) del certificado de la CA, se pueden obtener los siguientes resultados:

- Reinstalación correcta (Figuras C.45 y C.46).
- Error en la instalación de certificados (Figuras C.47 y C.48).
- Error en la generación de claves (Figuras C.49 y C.50).
- Error en la solicitud de reinstalación (Figuras C.51 y C.52).

Figura C.45: Notification Android: Reinstalación correcta de certificados

Figura C.46: Ticker Android: Reinstalación correcta de certificados

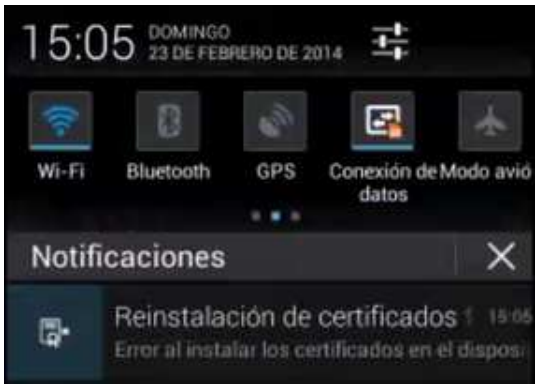

Figura C.47: Notification Android: Error en la reinstalación de certificados

Figura C.48: Ticker Android: Error en la reinstalación de certificados

Figura C.49: Notification Android: Error en la generación de claves en rollover

Figura C.50: Ticker Android: Error en la generación de claves en rollover

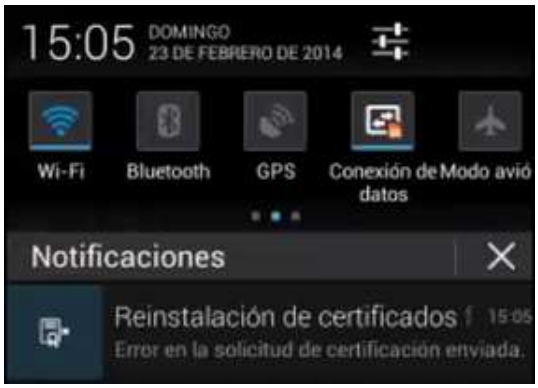

Figura C.51: Notification Android: Error en la solicitud de rollover

Figura C.52: Ticker Android: Error en la solicitud de rollover

De igual manera, si se pulsa en la notificación de reinstalación tras caducidad (Figuras C.33 y C.34) o revocación (Figuras C.31 y C.32) única del certificado del dispositivo, se pueden obtener los siguientes resultados:

- Reinstalación correcta (Figuras C.53 y C.54).
- Error en la instalación de certificados (Figuras C.55 y C.56).
- Error en la generación de claves (Figuras C.57 y C.58).
- Error en la solicitud de reinstalación (Figuras C.59 y C.60).

Figura C.53: Notification Android: Reinstalación correcta de certificado de dispositivo

Figura C.54: Ticker Android: Reinstalación correcta de certificado de dispositivo

Figura C.55: Notification Android: Error en la reinstalación de certificado de dispositivo

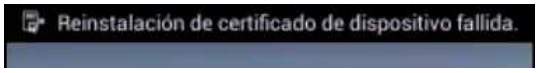

Figura C.56: Ticker Android: Error en la reinstalación de certificado de dispositivo

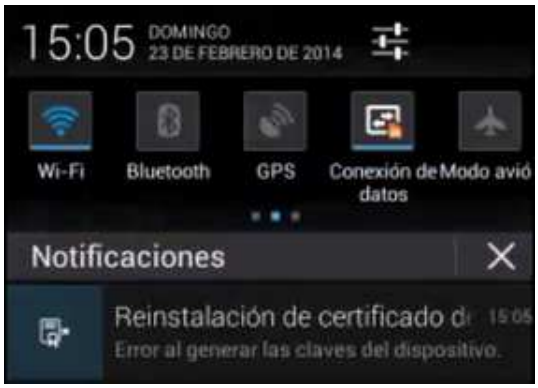

Figura C.57: Notification Android: Error en la generación de claves en renovación

Figura C.58: Ticker Android: Error en la generación de claves en renovación

Figura C.59: Notification Android: Error en la solicitud de rollover

Figura C.60: Ticker Android: Error en la solicitud de rollover